УНИВЕРЗИТЕТ У КРАГУЈЕВЦУ
ПРИРОДНО–МАТЕМАТИЧКИ ФАКУЛТЕТ

Татјана Стојановић

Развој и анализа метахеуристичких метода за испитивање задовољивости формула у вероватносним логикама

докторска дисертација

КРАГУЈЕВАЦ, 2015

I. Аутор
Име и презиме: Татјана Стојановић
Датум и место рођења: 29. јул 1974.
Садашње запослење: асистент на Природно-математичком факултету у Крагујевцу
II. Докторска дисертација
Наслов: Развој и анализа метахеуристичких метода за испитивање задовољивости формула у вероватносним логикама
Број страница: 110
Број слика: 13
Број библиографских података: 122
Установа и место где је рад израђен: Природно-математички факултет, Крагујевац
Научна област (УДК): Вештачка интелигенција (004.8)
Ментор: Зоран Огњановић
III. Оцена и одбрана
Датум пријаве теме: 18. јануар 2013.
Број одлуке и датум прихватања докторске дисертације:
Комисија за оцену подобности теме и кандидата: Зоран Огњановић, Татјана Давидовић, Милош Ивановић
Комисија за оцену докторске дисертације: Зоран Огњановић, Татјана Давидовић, Небојша Икодиновић, Милош Ивановић
Комисија за одбрану докторске дисертације: Зоран Огњановић, Татјана Давидовић, Небојша Икодиновић, Милош Ивановић
Датум одбране дисертације:

Садржај

Сажетак

У овој докторској дисертацији су на почетку дате основе вероватносних и дифолтних логика и основе оптимизационог приступа решавању различитих врста проблема. Вероватносне логике представљене у дисертацији су LPP_2, LPP_2^{ext} и $LPCP$ које, уз дифолтну логику, дају добар основ за представљање непрецизног и непотпуног знања, као и за резоновање над тим знањем. Од постојећих оптимизационих метода, за представљање су издвојени метахеуристички приступи који су већ коришћени за решавање проблема задовољивости формула у логици LPP_2^{ext}. Поред ових метода, приказана је метода оптимизације колонијом пчела (BCO) која је до сада дала добре резултате у решавању различитих комбинаторних проблема, а сада је по први пут употребљена као основна метода у решавању проблема задовољивости формула. Такође, до сада је BCO метода коришћена за решавање проблема за које постоји неко решење, а примена методе је имала за циљ да то решење поправи. Сада се по први пут BCO метода користи за проналажење решења.

Главни делови тезе посвећени су примени методе оптимизације колонијом пчела на проблем задовољивости формула логике $LPCP$ и примени на дифолтно резоновање. За проблем задовољивости формула $LPCP$ логике до сада није постојао аутоматски доказивач теорема. Сва постојећа решења, која се баве проблемом задовољивости формула у вероватносним логикама, су разматрала логике са операторима апсолутне вероватноће, при чему су вероватноће имале вредности из скупа реалних бројаве. Први пут је развијен аутоматски доказивач теорема који прихвата формуле са условним вероватноћама и, додатно, дозвољава да вредности вероватноћа припадају Хардијевом пољу бројева $Q[\varepsilon]$.

У дисертацији је детаљно представљен начин свођења проблема задовољивости формула у логици $LPCP$ на проблем линеарног програмирања, као и примена BCO методе за решавање добијеног линеарног проблема. BCO метода је изабрана на основу искуства да се бољи резултати добијају коришћењем метода базираних на популацијама, захваљујући чињеници да оне дозвољавају деградацију квалитета решења. Са друге стране, решења базирана на локалној претрази врло лаку упадну у замку локалног минимума, што често не води проналажењу решења. У раду су приказани и анализирани резултати добијени тестирањем. За потребе поређења резултата коришћен је и егзактан (директан) приступ у решавању добијеног проблема линеарног програмирања помоћу Фурије-Моцкин методе елиминације (FME). Добијени резултати су показали велику предност хеуристичког приступа над директним

приступом како у успешности тако и у времену потребном за извођење закључака.

Примена BCO методе на дифолтно резоновање захтевала је прилагођавање развијеног алгоритма и нов приступ у представљању бројева из Хардијевог поља бројева $Q[\varepsilon]$. У циљу повећања ефикасности у извођењу закључака у дифолтном резоновању, било је неопходно тестирати различите стратегије, а добијени резултати су детаљно приказани у дисертацији. У случају дифолтног резоновања добијени резултати су, такође, поређени са резултатима добијеним применом FME методе, при чему је, поново, показана велика надмоћ хеуристичког приступа над директним приступом. Приказана метода представља први хеуристички приступ дифолтном резоновању. Сви до сада развијани доказивачи су се заснивали на различитим директним приступима.

Summary

At the beginning of this thesis basics of probabilistic and default logics are given, as well as basics of optimization approach to solving different types of problems. Probabilistic logics presented are LPP_2, LPP_2^{ext} and $LPCP$. Together with default logic they make a good base for presenting imprecise and incomplete knowledge and reasoning above this knowledge. Emphasis is given on meta-heuristic approaches, that have already been used for solving satisfiability problem in LPP_2^{ext}. Beside these methods, Bee Colony Optimization method (BCO) is presented. It previously gave good results solving hard combinatorial problems, this being the first time it is used as basic method for dealing with formula satisfiability problems. Also, all previous uses of BCO consisted improving the some feasible solution of the given problem. This is the first implementation of BCO method for finding solutions.

Main parts of thesis are dedicated to implementation of BCO on solving satisfiability problem for $LPCP$ logic, and BCO implementation on default reasoning. Satisfiability problem for $LPCP$ logic had no automated solver before this thesis. All of the existing solutions dealing with satisfiability problem in probabilistic logics considered logics with absolute probability operators, with real value probabilities. This is the first time that a solver that accepts formulas with conditional probabilities has been presented. Additionally, it allows probability values to be from Hardy fields $Q[\varepsilon]$.

Thesis presents, in detail, the way of reducing satisfiability problem in $LPCP$ logic to linear programming problem, as well as application of BCO method for solving the resulting linear problem. BCO method is chosen based on experience of roviding better results then the using evolutionary methods, due to the fact that it allow solution quality degradation. On the other hand, solutions based on local search easily reach local optima, which often prevents the system from finding solutions. Thesis shows and analyzes testing results. For the purposes of results comparison, direct approach to solving resulting linear programming problem, Fourier-Motzkin elimination method (FME) was used. Results show great advantage of heuristic approach over FME, both regarding the success rate and time required for reaching conclusion.

Application of BCO to default reasoning required adjusting the developed algorithm, as well as new approach to representing numbers from Hardy's field $Q[\varepsilon]$. In order to improve efficiency in default reasoning testing of different strategies was required. This thesis shows detailed results of these tests. In case of default reasoning, results were also

compared to results of FME method. Again, great advantage of heuristic approach was demonstrated. The method used presents the first heuristic approach to solving problems in default reasoning. All previous solvers were based on different direct approach methods.

Глава 1

Увод

О машинама које мисле и вештачким бићима прича се почев од грчких митова. Машине са људским обликом правили су мајстори у свим великим цивилизацијама, укључујући Јан Шија (енг. *Yan Shi*) у III веку пре нове ере, Херона Александријског (енг. *Hero of Alexandria*) почетком нове ере и Ал-Јазарија (енг. *Al-Jazari*) који је живео крајем XII века. Људи су веровали да ће откривањем праве природе богова бити у стању да их репродукују. Данас, у ери убрзаног развоја рачунара, покушаји да се разуме природа и начин људског мишљења воде развоју вештачке интелигенције. Овом проблему се приступа са различитих становишта, зато и постоји толико различитих дефиницаја вештачке интелигенције. Једни је описују кроз системе који мисле као људи, други као системе који мисле рационално, трећи као системе који делују као људи, четврти као системе који делују рационално [104]. Без обзира на изабран приступ, рачунар који поседује вештачку интелигенцију мора да буде у стању да складишти оно што зна и чује (представља знање) и да користи ускладиштене информације ради давања одговара на питања и извођења нових закључака (аутоматско расуђивање). За бављење овим сегментом вештачке интелигенције најбоље оквире пружа математичка логика, а поготову комбинација математичке логике и теорије вероватноће, која треба да омогући рад са непрецизним и непотпуним информацијама.

1.1 Развој математичке логике и теорије вероватноће

Формално резоновање има јако дугу историју. Кинески, индијски и грчки филозофи су пре нове ере поставили основе дедуктивног размишљања које је потом развијано вековима. Са друге стране, игре на срећу иницирале су развој вероватноће. Ове две математичке дициплине, математича логика и вероватноћа, су се кроз историју често комбиновале: заснивање теорије вероватноће на логичким основама, пручавање логика у којима се истинитосна вредност формула израчунава помоћу функција које су вероватносне мере или личе на њих, проширивање логичког језика конструкцијама које омогућавају да се непосредно говори о вероватноћи итд.

Након дуже стагнације у развоју математичких дисциплина, у седамнаестом веку Лајбниц (нем. *G.ottfried Wilhelm von Leibniz*, 1646 – 1716) својим радовима прави огроман помак и поставља основе за даљи развој, пре свега, анализе, нестандардне анализе, математичке логике, вероватноће. Лајбниц је био инспирисан радом Паскала (фран. *Blaise Pascal*, 1623 – 1662), који је патентирао „машину за рачунање" и радио је на развоју универзалне науке (лат. *Scientia universalis*) која је требало да омогући аритметизацију закључивања. Лајбниц је разматрао вероватносну логику као алат за процену несигурности и за дефинисање вероватноће као мере знања. У својим есејима [64, 65, 66], Лајбниц је предложио да се алат развијен за анализу игара на срећу, примени и на развој нове логике која би се бавила степенима вероватноћа и која се може користити за доношење рационалних одлука у присуству контрадикторних чињеница. Он је разликовао две врсте рачуна. Први, рачун унапред, се бавио проценом вероватноће догађаја ако је позната вероватноћа услова. Други, рачун уназад, је процењивао вероватноћу узрока, уколико је вероватноћа последице позната. Лајбниц је имао пуно наследника, међу којима су најзначајнији, када је у питању вероватносна логика браћа Јакоб и Јохан Бернули (нем. *Jacob Bernoulli*, познат и као *James* или *Jacques*, 1655 – 1705 и нем. *Johann Bernoulli*, 1667 – 1748), Бајес (енг. *Thomas Bayes*, 1702 – 1761), Јохан Ламберт (фран. *Jean-Henri Lambert*, 1728 – 1777), Лаплас (фран. *Pierre Simon de Laplace*, 1749 – 1827), Бернард Болцано (чеш. *Bernhard Placidus Johann Nepomuk Bolzano*, 1781 – 1848), Де Морган (енг. *Augustus De Morgan*, 1806 – 1871), Бул (енг. *George Boole*, 1815 – 1864), Вен (енг. *John Venn*, 1834 – 1923), Мек Кол (енг. *Hugh MacColl*, 1837 – 1909), Пирс (енг. *Charles S. Peirce*, 1839 – 1887), Порецки (рус. *Платон Сергеевич Порецкий*, 1846 – 1907) и други.

Јакоб Бернули је први који је унапредио Лајбницове идеје. Он је предложио процедуру за одређивање нумеричког степена поузданости претпоставке добијене на основу аргумената. Реч аргумент је коришћена да означи исказе и релације импликације између претпоставки и закључака. Бернули се бавио и питањем израчунавања степена поузданости када постоји више аргумената за исти закључак.

Ламберт је у својој књизи [62] анализирао силогистичка извођења облика „ако је B 3/4 од A и C је A, тада је C B са вероватноћом 3/4". У Бајесовом раду [4] се први пут појављују резултати везани за условне вероватноће. Записано модерном нотацијом, Бајес је разматрао проблем одређивања вероватноће $P(A|B)$ где је A исказ „$P(E) \in [a, b]$", док је B исказ „догађај E се реализовао p пута и није се реализовао q пута у $p + q$ независних покушаја". Болцано [6] је логику посматрао као основну теорију науке, док је вероватноћа била део логике. Он је посматрао ваљаност исказа $A(x)$ као меру скупа $\{c : \models A(c)\}$, тј. као $\frac{|\{x:x\in U\wedge U\models A(x)\}|}{|\{x:x\in U\}|}$. Релативна ваљаност је релација међу исказима и има исте особине као условна вероватноћа у данашњим терминима. Болцано је доказао више теорема које се односе на релативну ваљаност. Де Морган се у својој књизи [18] бавио вероватносним закључивањем дајући подршку нумеричком вероватносном приступу као делу логике. Уместо систематичног приступа, Де Морган је описивао неке проблеме и покушавао да их реши логичким приступом. Интересантно је да је Де Морган направио неке грешке при решавању

проблема, које су углавном произилазиле из игнорисања (не)зависности догађаја.

Рачун који је Бул предложио [7, 8] иницирао је убрзани развој математичке логике. Бул је настојао да направи систем који би био основа логичког рачуна као и опште методе за примену у теорији вероватноће. Бул је тврдио да најопштији проблем у теорији вероватноће може да се реши уколико се посматра произвољан скуп логичких функција $\{f_1(x_1,\ldots,x_m),\ldots,f_k(x_1,\ldots,x_m),F(x_1,\ldots,x_m)\}$ и одговарајуће вероватноће $p_1 = P(f_1(x_1,\ldots,x_m)),\ldots,p_k = P(f_k(x_1,\ldots,x_m))$ са циљем да се одреди $P(F(x_1,\ldots,x_m))$ у зависности од $p_1,\ldots,p_k$. Он је објаснио везу између логике са класичним везницима и особина вероватноћа догађаја. Бул се ограничио на ексклузивну дисјункцију и веровао је да се сваки подисказ може изразити у терминима основних и независних компоненти. На тај начин, вероватноћа или-конструкције је једнака збиру вероватноћа компонената, док је вероватноћа и-конструкције једнака производу вероватноћа компонената. На тај начин је могуће трансформисати логичке функције догађаја у систем алгебарских функција одговарајућих вероватноћа. Бул је решавао овако добијене системе процедуром која је еквивалентна Фурије-Моцкин методи елиминације. Његова процедура, иако није потпуно успешна, пружила је основу за вероватносно резоновање. У [36, 37] је дата исправка Булове процедуре коришћењем приступа линеарним програмирањем.

Булови наследници су се трудили да унапреде Булове идеје. Један од њих је Порецки [97]. Пирс у [94] и Мек Кол у [79] су појаснили означавање условне вероватноће као шансе да је тврђење тачно, под претпоставком да је друго тврђење тачно и увели су одговарајућу ознаку x_a (која одговара ознаци $P(x|a)$).

Мек Кол је, истовремено са Фрегеом (нем. *Friedrich Ludwig Gottlob Frege*, 1848 – 1925), развијао исказни рачун као независну грану логике. Он је био први аутор који је, у [75], покушао да дво-вредносну логику прошири трећом истинитосном вредношћу. Добијени систем садржи сигурне, немогуће и променљиве исказе. Искази прва два типа су или тачни или нетачни, док је исказ трећег типа некада тачан, а некада нетачан.

Вен је 1870-их развијао идеју да повећа фреквенцију појаве појмова о вероватноћи у логици. Он је сматрао да је вероватносна логика, заправо, логика низа тврђења. Низ од једног елемента овог типа придружује датом исказу једну од две вредности 0 или 1, док бесконачан низ придружује ма који реалан број из интервала [0,1]. Неки традиционални логичари су били незадовољни укључивањем индукције у дефиниције појмова везаних за вероватноћу, али остали су наставили свој рад у овом правцу.

Током прве половине 20-ог века постојала су бар три правца у развоју теорије вероватноће. Истраживачи који су припадали првом правцу, Ричард фон Мис (нем. *Richard von Miss*, 1883 – 1953) и Ханс Рајхенбах (нем. *Hans Reichenbach*, 1891 – 1953) су, на пример, посматрали вероватноћу као релативну фреквенцу и извели правила на основу те интерпретације. Други приступ је карактеристичан по развоју формалног рачуна за вероватноћу. Неки аутори из тог правца су Џеорџ Болман (нем.

Georg Bohlmann, 1869 – 1928), Сергеј Натанович Берштејн (рус. *Сергей Натанович Бернштейн*, 1880 – 1968) и Емил Борел (фран. *Emile Borel*, 1871 – 1956). Овај правац је кулминирао Колмогоровом (рус. *Андрей Николаевич Колмогоров*, 1903 – 1987) аксиоматизацијом вероватноће [58]. На крају, неки истраживачи, попут Џона Кејнса (енг. *John M. Keynes*, 1883 – 1946), Рајхенбаха и Рудолфа Карнапа (нем. *Rudolf Carnap*, 1891 – 1970) су наставили Булов приступ повезујући вероватноћу и логику.

У Кејнсовим истраживањима, вероватноћа је посматрана као основни појам који се не дефинише. Он је предложио аксиоматску анализу релација међу исказима који се понашају као условне вероватноће. Са становишта савремене логике, овај аксиоматски систем није прихватљив, пошто у њему, на пример, не постоје спецификација синтаксе, правила извођења и слично.

Карнапов рад на логичким основама вероватноће је био покушај да се развије логички појам вероватноће. Карнап је повезао појмове индуктивног извођења, вероватноће и потврђивања. Он је био међу првим истраживачима који је јасно разликовао два различита појма вероватноће. Појам вероватноће као релативне фреквенције, која је коришћена у статистичким истраживањима, је емпиријске природе и из тог разлога непогодна за развој индуктивне логике. Карнапу је за развој индуктиве логике, која је за њега била исто што и вероватносна логика, био потребан логички појам вероватноће као степена потврде неке хипотезе на основу неких доказа, тј. логичка релација између два исказа, означена са $c(h, e)$. Он је фиксирао унарни језик првог реда за исказивање h и e, да би могао да проучава особине релације c. Иако овај рад није био у потпуности успешан, он је подстакао развој вероватносне логике првог реда [26, 27, 108, 114].

Рајхенбах је истраживао логичке структуре вероватносних исказа са филозофског и техничког аспекта. Он је увео основну вероватносну релацију између класа и реалних бројева користећи формулу облика $P(A, B) = p$ у значењу „за свако i, ако x_i припада класи A, тада y_i припада класи B са вероватноћом p".

Алекасндар Крон (1928 – 2000) је проучавао везу између вишевредносних логика и теорије вероватноће [60]. Он је посматрао унарну операцију генерисања Булове алгебре скупа формула и вероватносну функцију дефинисану на тој алгебри. При томе је дао неколико тврђења којима се повезују теорија вероватноће (условна вероватноћа, независност) и логика (импликација, доказ).

Упркос радовима Рајхенбаха, Карнапа и њихових следбеника, током друге половине 20-ог века развој логике и теорије вероватноће су текли потпуно независно. Крајем 19-ог века, независно од алгебарског приступа, развој математичке логике је био инспирисан потребом логичког заснивања математике. Главни представник тог правца је био Фреге, који се трудио да објасни основне логичке везе између појмова и математичких исказа. Фреге је увео истинитосне вредности, као посебне врсте апстрактних вредности, чиме је сваки исказ именован као тачан или нетачан. Јасно је да, по Фрегеу, истинитосне вредности имају посебан статус које никако нису повезане са вероватноћом. Овакав приступ је достигао врхунац Геделовим (нем. *Kurt*

Friedrich Gödel, 1904 – 1977) радом, који је доказао потпуност логике првог реда [31]. Овим радовима логика првог реда је заузела централно место у заједници логичара и тек крајем 70-их година XX века се поново јавио интерес за вероватносну логику.

Најважнији помак у вероватносној логици, након Лајбница и Була, направио је Кислер (енг. *Howard Jerome Keisler*, 1936 –). Сврха његовог познатог рада [53] је била да понуди модел-теоријски приступ теорији вероватноће. Овај рад је, такође, омогућио коришћење нестандардне анализе као корисног метода. Кислер је увео неколико корисних логичких квантификатора, попут $Px > r$. Формула $(Px > r)\phi(x)$ означава да скуп $\{x : \phi(x)\}$ има вероватноћу већу од r. Рекурзивну аксиоматизацију ове логике, означаване са L_{AP}, дао је Хувер (енг. *Douglas N. Hoover*) [49]. Својим радовима Кислер и Хувер су направили велики допринос на овом пољу. Доказали су теорему потпуности за разне врсте модела и многе друге модел-теоријске теореме. Развојем вероватносне теорије модела омогућено је проучавање логика веће изражајности.

Бајесов принцип вероватносног расуђивања је почео рано да се користи, поготово у креирању експертних система за медицинску дијагностику. Међутим, он није имао теоријски модел, па је из тог разлога имао бројне недостатке. Овакви модели су почетком 1970-их престали да се користе за вероватносно резоновање. Средином 1980-их је порастао интерес за вероватносне логике због развоја многих области примене резоновања са непоузданим знањем, попут економије, вештачке интелигенције, рачунарске науке, филозофије и тако даље.

У наставку ће кратко бити приказани разлози за развој хеуристичких метода за решавање различитих проблема, а затим ће бити дат опис проблема који је решаван и који представља централни део ове дисертације. Осим тога биће дате и полазне претпоставке у решавању описаног проблема као и преглед постојећих решења сличних проблема.

1.2 Метахеуристике

Развојем рачунара и потребом да се решавају проблеми код којих је простор стања који треба претражити огроман, јавља се потреба за развојем различитих метода оптимизације решавања проблема. Најопштије гледано, методе оптимизације се могу поделити на егзактне и приближне, које се даље деле на хеуристичке и симулационе [15]. Егзактне методе су најпожељније, обзиром да гарантују добијање оптималног решења, али су, због велике комплексности, применљиве на мали број проблема. Са друге стране, симулационе методе могу да се примењују на јако велики број проблема, али је њихово решење најнепоузданије. Код решавања проблема код којих је број (допустивих) решења коначан, али изузетно велики, велики успех је показала примена хеуристичких метода.

Основна предност хеуристичких метода је релативно велика брзина којом проналазе решења. Добијена решења нису увек оптимална, чак често не постоји ни оцена

квалитета добијеног решења. Међутим, обзиром да реални проблеми захтевају проналажење решења у што краћем временском периоду, решења која дају хеуристичке методе могу бити довољно добра. Са друге стране, када неко решење проблема већ постоји, могу се применити разне технике за његово побољшање, што је основна идеја приликом настајања хеуристика.

У решавању неког проблема, наравно, могу се комбиновати различити типови хеуристика. Хеуристике су се током времена показале као једини практични начин за решавање различитих оптимизационих проблема, па се, сходно томе, овоме посвећује све већа пажња. То је довело до развоја универзалних хеуристика или тзв. метахеуристика. Класичне хеуристике су, углавном, биле намењене решавању конкретних проблема и користиле су познате особине датог проблема за његово решавање. Метахеуристике се састоје од уопштених скупова правила која се могу применити за решавање разноврсних проблема. Метахеуристички приступи засновани су на општим алгоритмима оптимизације који подразумевају примену итеративних поступака у циљу поправљања неког постојећег решења.

Многе метахеуристичке методе су се развиле по угледу на процесе у природи, на пример генетски алгоритми (енг. *Genetic Algorithm*, ГА), разни еволутивни алгоритми (енг. *Evolutionary Algorithm*, EA), неуралне мреже (енг. *Neural Network*), колоније мрава (енг. *Ant Colony Optimization*, ACO), колоније пчела (енг. *Bee Colony Optimization*, BCO и енг. *Bee Swarm Optimization*, BSO) и други. Неке су инспирисане локалним претраживањем (енг. *local search*) са разним идејама да се избегне замка локалног минимума, на пример вишестартно локално претраживање (енг. *Multistart local search*, MLS), метода променљивих околина (енг. *Variable neighborhood search*, VNS), табу претраживање (енг. *Tabu-search*, TS), похлепна стохастичка прилагодљива процедура претраживања (енг. *Greedy Randomized Adaptive Search Procedure*, GRASP) и друге.

Скорашња истраживања су показала да метахеуристичке методе дају добре резултате у претраживању знања у домену вештачке интелигенције. Једна таква примена биће тема овог рада.

1.3 Поставка проблема

Као што је већ напоменуто, вероватносне логике дају добар оквир за представљање знања и резоновање у ситуацијама када су чињенице непоуздане, контрадикторне или непотпуне. Немонотоно резоновање боље осликава закључивање у реалним ситуацијама, а вероватносне логике су једна од могућности за реализацију овакве врсте закључивања. У овом приступу се врши проширивање класичне логике изразима који омогућавају да се говори о вероватноћи, док формуле остају истините, односно лажне. На тај начин се ствара могућност да формално изразимо реченице попут "условна вероватноћа од A, под условом B, је барем s". У модел-теоријском

и доказно-теоријском проучавању вероватносних логика уобичајено је да се разматрају различите врсте вероватноћа (са коначним, пребројивим или непребројивим кодоменима, коначне или сигма-адитивне итд.) и вероватносних оператора (оператори апсолутне вероватноће, оператори условне вероватноће, оператори погодни за исказивање прецизних или приближних вероватноћа, итд.) помоћу којих се формирају формални логички искази.

Домени примене овако представљеног знања су бројни. Најочигледнији пример је дијагностиковање у медицини, где присуство или одсуство неког симптома указује на неку болест сасвим сигурно или у одређеном проценту. У раду ће бити представљен и начин моделовања дифолтног закључивања користећи вероватносну логику са приближним условним вероватноћама [101]. Дифолтно резоновање представља још једну врсту немонотоног резоновања погодног за формализацију људског резоновања код ког се често, у случају пристизања нових информација, врши ревизија претходних одлука. Осим, у већ поменутој медицини, дифолтно резоновање може да нађе своју примену у закључивањима која се тичу правних регулатива, спецификацији различитих система и апликација и слично.

Највећи део овог рада биће посвећен вероватносној логици $LPCP$ са приближним условним вероватноћама [101]. Кодомен за посматране вероватноће је Хардијево поље $Q(\varepsilon)$, где је ε инфинитезимално мали број. Осим претстављања аксиоматског система, његове коректности и комплетности, посебна пажња биће посвећена проблему задовољивости обзиром на могуће примене. По угледу на проблем задовољивости класичних формула (енг. *satisfiability problem*, SAT), који је као основни NP-комплетни проблем постао централни проблем теорије сложености израчунавања, проблем задовољивости вероватносних формула се означава са PSAT, а, конкретно, проблем задовољивости у логици $LPCP$ биће означен са CPSAT-ε.

Вероватносна логика са приближним условним вероватноћама представља проширење исказног рачуна. Вероватносне формуле добијене на овај начин су облика $CP_{\geqslant s}(\alpha, \beta)$, $CP_{\leqslant s}(\alpha, \beta)$ и $CP_{\approx s}(\alpha, \beta)$, са значењем „вероватноћа да је α под условом да је β је најмање s", „највише s" и „приближно s", респективно. Проблем задовољивости у овој логици се може свести на проблем линеарног програмирања [101]. Решавање добијених система стандрадним методама је због комплексности проблема практично неприменљиво у реалним ситуацијама. Примена, на пример, Фурије-Моцкин методе елиминације (енг. *Fourier-Motzkin elimnation method*, FME) доводи до експонецијалног раста броја неједнакости у систему. Из овог разлога, за решавање ове класе проблема добре резулатате даје примена различитих метахеуристичких метода.

Посебна пажња посвећена је резоновању у дифолтној логици, која, такође, представља проширење исказне логике дифолтним правилима. Дифолтна правила се могу представити формулама облика $CP_{\approx 1}(\alpha, \beta)$, а добијени формални систем се у потпуности поклапа са дифолтним системом **P**. На овај начин се дифолтно резоновање своди са специјални случај резоновања у вероватносној логици са приближним условним вероватноћама. Обзиром да је главни циљ дифолтног резоновања

испитивање да ли је могуће извести закључак B из претпоставки $A_1, A_2, \ldots, A_n$, ово резоновање ће бити сведено на испитивање задовољивости следећа три скупа формула: $\Delta = \{A_1, A_2, \ldots, A_n\}$, $\Phi_1 = \Delta \cup \{B\}$ и $\Phi_2 = \Delta \cup \{\neg B\}$, где су $A_1, A_2, \ldots, A_n, B$ дифолтна правила. Задовољивост скупа Δ потврђује конзистентност дифолтне базе. Задовољивост скупа Φ_1 показује да дифолт B није у контрадикцији са базом. Из незадовољивости скупа Φ_2 се изводи закључак да је B последица скупа претпоставки.

1.4 Поставка основне хипотезе

Предмет овог рада је развој метода за испитивање задовољивости формула у вероватносним логикама са операторима условне вероватноће и испитивање задовољивости скупа дифолта моделираним уз помоћ ових логика. Обзиром на сложеност посматраних проблема њихово решавање потпуним процедурама могуће је само за примере веома малих димензија. Ова тврдња биће поткрепљена резултатима добијеним применом FME методе за решавање система неједнакости добијених свођењем проблема задовољивости на проблем линеарног програмирања. FME метода биће примењена и у случају CPSAT-ε и у случају задовољивости скупа формула у дифолтним логикама. За CPSAT-ε до сада није постојао аутоматски проверавач задовољивости формула, док су за дифолтно резоновање развијане само егзактне (директне) методе.

Као пример бољег приступа у решавању ових проблема биће приказана метахеуристичка метода оптимизације колонијом пчела (енг. *Bee Colony Optimization*, BCO). BCO је метахеуристичка метода која је показала веома добре резулате у решавању различитих оптимизационих проблема. То је стохастичка метода, која се заснива на случајној претрази (енг. *random-search technique*) и припада класи алгоритама базираних на популацијама. Ова техника користи аналогију између начина на који пчеле у природи траже храну и оптимизационог алгоритма за тражење оптималног решења датог оптимизационог проблема. BCO метода је по први пут искоришћена за проналажење решења прблема. До сада, она је коришћена као оптимизациона метода која већ постојеће решење неког проблема покушава да поправи. Такође, искуство показује да коришћење метода базираних на популацијама даје боље резулате од метода које су засноване, на пример, на локалној претрази. Методе засноване на популацијама дозвољавају деградацију квалитета решења, док, насупрот њима, методе засноване на локалној претрази, лако упадну у замку локалног минимума, која не води проналажењу решења.

1.5 Преглед постојећих решења

За CPSAT-ε не постоји аутоматски доказивач теорема нити је BCO метода коришћена за решавање проблема задовољивости. Примена метахеуристичких метода за решавање проблема задовољивости, ипак, није нов приступ.

У литератури могу да се нађу хеуристички приступи у решавању проблема задовољивости у исказној логици (SAT) и проблема одређивања максималног броја клауза у датој формули у конјунктивној нормалној форми које могу бити задовољене при некој интерпретацији, такође, у исказној логици (енг. *maximum satisfiability problem*, MAX-SAT). Многа понуђена решења се заснивају на процедурама локалне претраге [55, 68, 109, 113]. Затим, постоје решења за SAT која се заснивају на интелигенцији популације, попут ACO и BSO представљених у [20, 119]. За SAT и MAX-SAT су у [77, 99] представљени приступи коришћењем GA, а комбинацијом GA са другим метахеуристичким методама је представљена у [82].

Вероватносне логике су представљене у радовима [23], [88] и [84]. За ове логике проблем задовољивости је означен са PSAT и за његово решавање је у [85, 86] представљен приступ коришћењем GA, у [50] коришћењем VNS, у [87] комбинација GA и VNS, коришћењем локалне претраге у [52], а коришћењем TS у [41]. Међутим, између посматраног PSAT и CPSAT-ε, који ће бити предмет овог рада, постоје две кључне разлике:

- CPSAT-ε се односи на рад са условним вероватносним оператором, за разлику од PSAT, где се разматрају апсолутни вероватносни оператори;
- вероватноће у формулама у CPSAT-ε могу имати инфинитезималне вредности, док су код PSAT све вероватноће реално вредносне.

За закључивање у дифолтним логикама у литератури не постоје предложени хеуристички приступи. Неки алгоритми за резоновање у условним базама знања и њихова комплексност су дати у [22], а сличан проблем је разматран и у [33]. Условна база знања је пар $KB = (L, D)$, где је L коначан скуп исказних формула и D коначан скуп дифолта. За систем **P** је показано да је проблем испитивања да ли је условна база знања конзистентна, NP-комплетан у општем случају и P-комплетан у случају када се ради са хорновским формулама. У радовима [69, 70, 71] су представљени различити формални системи који се састоје од фомула исказне логике, дифолта и вероватносног знања записаног у облику условних ограничења. Ови формализми представљају приступ резоновању над статистичким и субјективним знањем. У овим радовима дати су односи између представљених формализама, алгоритми за резоновање у датим системима и комплексност тих алгоритама. Детаљан приказ аутоматског дифолтног резоновања је дат у [19]. Додатно, алгоритами за дифолтно резоновање базирани на методи таблоа су приказани у [102, 107]. Аутоматски доказивач теорема за дифолтно резоновање заснован на програмском језику Пролог (енг. *Prolog*) је приказан у [105].

1.6 Преглед садржаја дисертације

У наредним поглављима овог рада биће приказани приступи у решавању проблема задовољивости у вероватносној логици са приближним условним вероватно-

ћама и дифолтној логици.

У поглављу 2 биће дат детаљан опис синтаксе, аксиоматско заснивање вероватносних логика, а посебна пажња биће посвећена логици $LPCP$, обзиром да су у наставку за ову логику развијани алгоритми за резоновање. Поглавље 3 садржи приказ дифолтног резоновања. детаљно је приказан систем **P**, као и његова веза са логиком $LPCP$. У поглављу 4 су дате основне дефиниције проблема оптимизације и преглед метахеуристичких метода које су до сада примењиване за решавање проблема резоновања у вероватносним логикама. Поред овх метода дат је детаљан опис методе оптимизације колонијом пчела као методе која је у наставку коришћена за развој алгоритама за резоновање у $LPCP$ и дифолтној логици.

Поглавља 5 и 6 предствљају централни део дисертације. Поглавље 5 садржи детаљан опис примене BCO методе на испитавње задовољивости формула $LPCP$ логике, затим анализу резултата добијених тестирањем, као и резултате добијене паралелизацијом развијеног алгоритма. У поглављу 6 су описани сви кораци у којима је било неопходно извршити модификацију BCO методе развијене за резоновање у $LPCP$ логици, да би добијени алгоритам дао прихватљиве резултате приликом примене на резоновање у дифолтној логици. Ово поглавље садржи и детаљнију анализу резултата добијених применом различих статегија, као и резултате тестирања применом најефикасније стратегије на примере из литературе.

Закључна разматрања и правци даљег развоја дати ду у поглављу 7, док су описи, раније познатих метода коришћених у раду дати у додатку А.

1.7 Захвалност

Велику помоћ приликом истраживања и изради дисертације пружили су ми професори Зоран Огњановић и Татјана Давидовић. На корисним саветима добијеним приликом рада на истраживању које је резултовало дисертацијом, захваљујем се професорима Небојши Икодиновић и Милошу Ивановићу.

Посебну захвалност за пружену помоћ, подршку и показано стрпљење током рада на дисертацији дугујем Дарку.

Глава 2

Вероватносне логике

2.1 Логика LPP_2

У овом делу биће описана синтакса и неке класе модела за логику LPP_2. Такође ће бити дата бесконачна аксиоматизација. Теореме о потпуности и комплетности логике LPP_2 су дате у раду [89].

2.1.1 Синтакса

Нека је $\mathbb{Q}[0,1]$ скуп свих рационалних бројева из $[0,1]$. Језик логике LPP_2 се састоји од

- пребројивог скупа исказних слова $\mathrm{Var} = \{p, q, r, \ldots\}$,
- класичних оператора $\neg$ и $\wedge$,
- и низа вероватносних оператора $P_{\leqslant s}$ за свако $s \in \mathbb{Q}[0,1]$.

Скуп For_C класичних исказних формула над скупом Var се дефинише на уобичајен начин. Елементи скупа For_C ће бити означавани са α, β,... Ако $\alpha \in For_C$ и $s \in \mathbb{Q}[0,1]$, тада се $P_{\geqslant s}\alpha$ назива *основна вероватносна формула*. Скуп For_P свих вероватносних формула је најмањи скуп који:

- садржи све основне вероватносне формуле и
- затворен је за правила: ако $A, B \in For_P$ тада $\neg A, A \wedge B \in For_P$.

Формуле из скупа For_P ће бити означаване са A, B,... Нека је $For_{LPP_2} = For_C \cup For_P$. Формуле из скупа For_{LPP_2} ће бити означаване са Φ, Ψ, ...

Остали класични оператори су дефинисани на уобичајени начин: $\alpha \vee \beta =_{def} \neg(\neg\alpha \wedge \neg\beta)$, $\alpha \to \beta =_{def} \neg\alpha \vee \beta$ и $\alpha \leftrightarrow \beta =_{def} (\alpha \to \beta) \wedge (\beta \to \alpha)$ и слично фаормуле из скупа For_P, док ће $\alpha \wedge \neg\alpha$ и $A \wedge \neg A$ бити записиване као $\bot$. Други вероватносни оператори се уводе следећим дефиницијама:

- $P_{<s}\alpha =_{def} \neg P_{\geqslant s}\alpha$,
- $P_{\leqslant s}\alpha =_{def} P_{\geqslant 1-s}\neg\alpha$,
- $P_{>s}\alpha =_{def} \neg P_{\leq s}\alpha$,
- $P_{=s}\alpha =_{def} P_{\geqslant s}\alpha \wedge P_{\leqslant s}\alpha$.

Као што се може видети, није дозвољено мешање исказних и вероватносних формула, као ни угњеждавање вероватносних оператора. Тако, на пример формуле $\alpha \vee P_{\geqslant s}\beta$ и $P_{\geq s}P_{\geqslant r}\alpha$ не припадају скупу For_{LPP_2}. Са друге стране, формула $(P_{>r}\alpha \wedge P_{=s}(\alpha \to \beta)) \to P_{\leqslant t}\beta \in For_{LPP_2}$ и може се схватити као „ако је вероватноћа од α већа од r и β следи из α са вероватноћом s, тада вероватноћа од β није већа од t".

2.1.2 Семантика

Семантика за логику LPP_2 се заснива на приступу са могућим световима [89].

Дефиниција 2.1.1. *LPP_2-модел је структура $\mathbf{M} = \langle W, H, \mu, v\rangle$ где је:*

- *W непразан скуп објеката званих светови,*
- *H алгебра подскупова од W,*
- *μ коначна адитивна мера, $\mu : H \to [0,1]$ где:*
  - *за свако $X \in H$, $\mu(X) \geq 0$,*
  - *$\mu(W) = 1$, и*
  - *$\mu(X_1 \cup X_2) = \mu(X_1) + \mu(X_2)$, за све дисјунктне скупове X_1, $X_2 \in H$,*
- *$v : W \times \mathrm{Var} \to \{\mathrm{true}, \mathrm{false}\}$ функција која сваком свету $w \in W$ додељује истинитосну валуацију $v(w)$ за исказна слова. Истинитосна валуација $v(w)$ се проширује на исказне формуле на уобичајени начин.*

Ако је $\mathbf{M}$ LPP_2-модел и $\alpha \in For_C$ скуп $\{w : v(w)(p) = \mathrm{true}\}$ се означава са $[\alpha]_{\mathbf{M}}$. Ознака $\mathbf{M}$ може бити изостављена из записа $[\alpha]_{\mathbf{M}}$, тј. биће коришћена ознака $[\alpha]$ ако је јасно о ком моделу $\mathbf{M}$ се говори. LPP_2-модел $\mathbf{M} = \langle W, H, \mu, v\rangle$ је *мерљив* ако $[\alpha]_{\mathbf{M}} \in H$ за сваку формулу $\alpha \in For_C$. У наставку ће пажња бити посвећена класи свих мерљивих LPP_2-модела, у ознаци $LPP_{2,Meas}$.

Дефиниција 2.1.2. ***Релација задовољивости*** $\models \subset LPP_{2,Meas} \times For_{LPP_2}$ *је дефинисана следећим условима за сваки $LPP_{2.Meas}$-модел $\mathbf{M} = \langle W, H, \mu, v\rangle$:*

- *ако је $\alpha \in For_C$, $\mathbf{M} \models \alpha$ ако и само ако за свако $w \in W$, $v(w)(\alpha) = \mathrm{true}$,*

- $\mathbf{M} \models P_{\geqslant s}\alpha$ *ако и само ако* $\mu([\alpha]) \geqslant s$,
- *за* $A \in For_P$, $\mathbf{M} \models \neg A$ *ако и само ако* $\mathbf{M} \not\models A$,
- *за* $A, B \in For_P$, $\mathbf{M} \models A \wedge B$ *ако и само ако* $\mathbf{M} \models A$ *и* $\mathbf{M} \models B$.

Дефиниција 2.1.3. *Формула* $\Phi \in For_{LPP_2}$ *је* ***задовољива*** *ако постоји* $LPP_{2,Meas}$*-модел* $\mathbf{M}$ *такав да* $\mathbf{M} \models \Phi$*; формула* Φ *је* ***ваљана*** *ако за сваки* $LPP_{2,Meas}$*-модел* $\mathbf{M}$ *важи да* $\mathbf{M} \models \Phi$*; скуп формула* T *је* ***задовољив*** *ако постоји* $LPP_{2,Meas}$*-модел* $\mathbf{M}$ *такав да* $\mathbf{M} \models \Phi$ *за свако* $\Phi \in T$.

2.1.3 Аксиоматски систем

Скуп свих ваљаних формула се може окарактерисати следећом аксиоматском схемом:

1. Све For_C-инстанце исказних таутологија
2. $P_{\geqslant 0}\alpha$
3. $P_{\leqslant r}\alpha \to P_{<s}\alpha, s > r$
4. $P_{<s}\alpha \to P_{\leqslant s}\alpha$
5. $(P_{\geqslant r}\alpha \wedge P_{\geqslant s}\beta \wedge P_{\geqslant 1}(\neg(\alpha \wedge \beta))) \to P_{\geqslant \min(1, r+s)}(\alpha \vee \beta)$
6. $(P_{\leqslant r}\alpha \wedge P_{<s}\beta) \to P_{(r+s)}(\alpha \vee \beta),\ r + s \leqslant 1$

и правилима извођења

1. Из Φ и $\Phi \to \Psi$ се изводи Ψ.
2. Из α се изводи $P_{\geqslant 1}\alpha$
3. Из $A \to P_{\leqslant a - \frac{1}{k}}$, за свако $k \leqslant \frac{1}{s}$ и $s > 0$ се изводи $A \to P_{\geqslant s}\alpha$

Аксиоматски систем ће бити обележен са Ax_{LPP_2}.

Дефиниција 2.1.4. *Формула* Φ *је* ***изводива*** *из скупа формула* T *(у ознаци* $T \vdash_{Ax_{LPP_2}} \Phi$*) ако постоји највише пребројив низ формула* Φ_0, $\Phi_1, \ldots$, Φ, *тако да је свако* Φ_i *аксиома или формула из скупа* T *или је изведена из претходних формула правилом извођења.* ***Доказ*** *за формулу* Φ *из скупа формула* T *је одговарајући низ формула. Формула* Φ *је* ***теорема*** *(у ознаци* $\vdash_{Ax_{LPP_2}} \Phi$*) ако је изводива из празног скупа.*

Дефиниција 2.1.5. *Скуп формула* T *је* ***конзистентан*** *ако постоји бар једна формула из* For_C *и бар једна формула из* For_P *које нису изводиве из* T*, у супротном је* T ***неконзистентно***. *Конзистентан скуп формула* T *је* ***максимално конзистентан*** *ако важи:*

- *за свако* $\alpha \in For_C$, *ако* $T \vdash_{Ax_{LPP_2}} \alpha$, *тада* $\alpha \in T$ *и* $P_{\geqslant 1}\alpha \in T$ *и*
- *за свако* $A \in For_P$ *или* $A \in T$ *или* $\neg A \in T$.

Скуп формула T *је* ***дедуктивно затворен*** *ако за свако* $\Phi \in For_{LPP_2}$, *ако* $T \vdash_{Ax_{LPP_2}} \Phi$ *тада* $\Phi \in T$.

Класичне и вероватносне формуле у дефиницији 2.1.5 се не обрађују на исти начин, тј. није неопходно да за сваку класичну формулу α, α или $\neg\alpha$ припада максимално конзистентном скупу, као што се захтева за формуле из скупа For_P.

На основу аксиоме 1 класична исказна логика представља подлогику логике LPP_2. Лако се показује да сваки LPP_2-доказ има два дела (један од њих може бити празан). У први део доказа су укључене само исказне формуле, док се у другом делу користе само формуле из скупа For_P. Ова два дела су одвојена применом правила извођења 2. Обзиром да не постоји инверзно правило, током извођења доказа је могуће прећи са дела исказних формула на део са вероватносним формулама, али обрнуто није могуће. Из овога се изводи да логика LPP_2 представља проширење класичне исказне логике. Аксиоме 2 - 6 се односе на вероватносни аспект LPP_2 логике. На основу аксиоме 2 свака формула је задовољива у скупу светова чија је мера бар нула. Заменом формуле α формулом $\neg\alpha$ добија се формула $P_{\geqslant 0}\neg\alpha$. На основу дефиниције оператора $P_{\leqslant 1}$ добија се нови облик аксиоме 2:

2' $P_{\leqslant 1}\alpha$ $(=\ P_{\geqslant 1-s}\neg\alpha$ за $s = 1)$

Овај облик намеће да је свака формула задовољива на скупу светова чија је мера највише 1, и тиме дефинише горњу границу за вероватноће формуле у $LPP_{2,Meas}$-моделима. На сличан начин, аксиоме 3 и 4 су еквивалентне са:

3' $P_{\geqslant t}\alpha \to P_{>s}\alpha,\ t > s$

4' $P_{>s}\alpha \to P_{\geqslant s}\alpha$

респективно. Аксиоме 5 и 6 се односе на адитивност мере. На пример, у аксиоми 5, ако су скупови светова, у којима су формуле α и β задовољиве, дисјунктни, тада мера скупа светова, у којима је $\alpha \vee \beta$ задовољиво, представља суму мера појединачних скупова светова. Правило извођења 1 је класично правило модус поненса (лат. *modus ponens*). Правило извођења 2 се може посматрати као правило нецеситације у модалној логици, али се може применити само на класичне исказне формуле. Правило извођења 3 је једино бесконачно правило извођења у систему, тј. врши се извођење једног закључка на основу пребројивог скупа претпоставки. Ово правило одговара Архимедовој аксиоми за реалне бројеве и интуитивно се може схватити да ако је вероватноћа произвољно блиска s, тада је она бар s.

2.1.4 Коректност и потпуност

Коректност посматраног система следи из коректности класичне исказне логике и особина вероватносне мере.

Теорема 2.1.1. *Аксиоматски систем Ax_{LPP_2} је коректан у односу на класу $LPP_{2,Meas}$-модела.*

Доказ теореме дат је у раду [89].

Теорема 2.1.2 (Јака потпуност)**.** *Скуп формула T је конзистентан ако и само ако T има $LPP_{2,Meas}$-модел.*

Доказ теореме дат је у раду [89].

2.1.5 Одлучивост и комплексност

Обзиром да је за исказне формуле проблем испитивања задовољивости скупа формула познат, од интереса је само разматрање задовољивости скупа For_P формула.

Теорема 2.1.3. *Логика LPP_2 је одлучива.*

Доказ ове теореме је, такође, дат у раду [89] где је описана процедура свођења проблема задовољивости скупа формула на проблем линеарног програмирања. Ова процедура ће детаљно бити описана у делу 2.3 који се односи на логику са условним приближним вероватноћама, која представља једно од проширења LPP_2 логике.

Теорема 2.1.4. *Проблем $LPP_{2,Meas}$-задовољивости је NP-комплетан.*

Доказ теореме дат је у раду [89].

2.2 Проширење вероватносне логике LPP_2

Проширење логике LPP_2 представљено је у радовима [23] и [112] и означено са LPP_2^{ext}. Логике представљене у ова два рада имају исту синтаксу, а тиме и исту изражајност, међутим битну разлику представљају дати аксиоматски системи. Обзиром да приступ дат у раду [112] омогућава доказивање теореме јаке потпуности, овде ће бити изложен тај приступ.

2.2.1 Синтакса

Слично као језик логике LPP_2 и језик логике LPP_2^{ext} се састоји од:

- пребројивог скупа исказних слова $\mathrm{Var} = \{p, q, r, \ldots\}$,
- класичних оператора $\neg$ и $\wedge$ и
- симбола P^* [1].

Скуп For_C се дефинише и означава на исти начин као у логици LPP_2, док ће тежинске формуле скупа For_W бити означаване са $A, B, C, \ldots$ и дефинисане на следећи начин:

- примитивни тежински терм је израз облика $P^*(\alpha)$, где је $\alpha \in For_C$
- тежински терм је израз облика $a_1P^*(\alpha_1) + a_2P^*(\alpha_2) + \cdots + a_kP^*(\alpha_k)$, где су $a_1, a_2, \ldots, a_k \in \mathbb{Q}$, $\alpha_1, \alpha_2, \ldots \alpha_k \in For_C$ и $k \geqslant 1$
- основна тежинска формула је израз облика $t \geqslant s$, где је t тежински терм и $s \in \mathbb{Q}$
- основне тежинске формуле су тежинске формуле и ако су A и B тежинске формуле, тада су и $\neg A$ и $A \wedge B$ тежинске формуле.

Нека је $For_{LPP_2^{ext}} = For_C \cup For_W$. Формуле из скупа $For_{LPP_2^{ext}}$ ће бити означаване са Φ, Ψ, ...

На основу особина неједнакости уводе се следеће дефиниције:

- $t < s =_{def} \neg(t \geqslant s)$
- $t \leqslant s =_{def} -t \geqslant -s$
- $t > s =_{def} \neg(t \leqslant s)$
- $t = s =_{def} (t \geqslant s) \wedge \neg(t > s)$

Овако дефинисаном логиком се омогућава резоновање о вероватноћама. На пример, реченица *„вероватноћа p је мања од $\frac{1}{3}$ и p је бар два пута вероватније него q"* се записује као $(3P^*(p) < 1) \wedge (P^*(p) - 2P^*(q) \geqslant 0)$, односно, користећи синтаксу из рада [23] као $(3w(p) < 1) \wedge (w(p) - 2w(q) \geqslant 0)$.

[1] У раду [23] коршћен је симбол w, али да би се избегла забуна између овог симбола и симбола за ознаку света у вероватносном моделу, уведен је нови симбол

2.2.2 Семантика

Дефиниција модела се у овом проширењу уводи на исти начин као за логику LPP_2 [112].

Дефиниција 2.2.1. *LPP_2^{ext}-модел је структура $\mathbf{M} = \langle W, H, \mu, v\rangle$ где је:*

- *W непразан скуп објеката названих светови,*
- *H алгебра подскупова од W,*
- *μ коначна адитивна мера, $\mu : H \to [0,1]$ где:*
  - *за свако $X \in H$, $\mu(X) \geq 0$,*
  - *$\mu(W) = 1$, и*
  - *$\mu(X_1 \cup X_2) = \mu(X_1) + \mu(X_2)$, за све дисјунктне скупове X_1, $X_2 \in H$,*
- *$v : W \times \mathrm{Var} \to \{\mathrm{true}, \mathrm{false}\}$ функција која сваком свету $w \in W$ додељује истинитосне валуације $v(w)$ за исказна слова. Истинитосна валуација $v(w)$ се проширује на исказне формуле на уобичајени начин.*

Ознака $[\alpha]_{\mathbf{M}}$ има исто значење као и у логици LPP_2 и слично $LPP^{ext}_{2,Maes}$ означава класу свих мерљивих модела, при чему је модел мерљив ако $[\alpha]_{\mathbf{M}} \in H$ за сваку формулу $\alpha \in For_C$..

Дефиниција 2.2.2. ***Релација задовољивости*** $\models \subset \ LPP^{ext}_{2,Meas} \times For^{ext}_{LPP_2}$ *је дефинисана следећим условима за сваки $LPP^{ext}_{2.Meas}$-модел $\mathbf{M} = \langle W, H, \mu, v\rangle$:*

- *ако је $\alpha \in For_C$, $\mathbf{M} \models \alpha$ ако и само ако за свако $w \in W$, $v(w)(\alpha) = \mathrm{true}$,*
- *ако $\mathbf{M} \models a_1 P^*(\alpha_1) + \cdots + a_k P^*(\alpha_k) \geqslant s$ ако и само ако $\sum_{i=1}^{k} a_i \mu([\alpha_i]) \geqslant s$,*
- *ако $A \in For_W$, $\mathbf{M} \models \neg A$ ако и само ако $\mathbf{M} \not\models A$,*
- *ако $A, B \in For_W$, $\mathbf{M} \models A \wedge B$ ако и само ако $\mathbf{M} \models A$ и $\mathbf{M} \models B$.*

Дефиниција 2.2.3. *Формула $\Phi \in For_{LPP_2}$ је **задовољива** ако постоји $LPP^{ext}_{2,Meas}$-модел $\mathbf{M}$ такав да $\mathbf{M} \models \Phi$; формула Φ је **ваљана** ако за сваки $LPP^{ext}_{2,Meas}$-модел $\mathbf{M}$ важи да $\mathbf{M} \models \Phi$; скуп формула T је **задовољив** ако постоји $LPP^{ext}_{2,Meas}$-модел $\mathbf{M}$ такав да $\mathbf{M} \models \Phi$ за свако $\Phi \in T$.*

2.2.3 Аксиоматски систем

Аксиоматски ситем $Ax_{LPP_2^{ext}}$ за LPP_2^{ext} је дат следећом аксиоматском схемом:

1. Све For_C-инстанце исказних таутологија
2. Све вероватносне инстанце ваљаних формула које се односе на линеарне неједнакости
   (a) $x \geqslant x$
   (b) $(a_1x_1 + \cdots + a_kx_k \geqslant c) \Leftrightarrow (a_1x_1 + \cdots + a_kx_k + 0x_{k+1} \geqslant c)$
   (c) $(a_1x_1 + \cdots + a_kx_k \geqslant c) \Rightarrow (a_{j_1}x_{j_1} + \cdots + a_{j_k}x_{j_k} \geqslant c)$, где је $j_1, \ldots j_k$ пермутација бројева $1, \ldots, k$
   (d) $(a_1x_1 + \cdots + a_kx_k \geqslant c) \wedge (a'_1x_1 + \cdots + a'_kx_k \geqslant c') \Rightarrow ((a_1 + a'_1)x_1 + \cdots + (a_k + a'_k)x_k \geqslant (c + c'))$
   (e) $(a_1x_1 + \cdots + a_kx_k \geqslant c) \Leftrightarrow (da_1x_1 + \cdots + da_kx_k \geqslant dc)$, ако је $d > 0$
   (f) $(t \geqslant c) \vee (t \leqslant c)$, ако је t терм
   (g) $(t \geqslant c) \Rightarrow (t > d)$, ако је t терм и $c > d$, тј. $(t \leqslant c) \Rightarrow (t < d)$, ако је t терм и $c < d$
   (h) $(t > c) \Rightarrow (t \geqslant c)$, i.e. $(t < c) \Rightarrow (t \leqslant c)$ ако је t терм
3. Све инстанце формула за вероватносно закључивање
   (a) $P^*(\alpha) \geqslant 0$
   (b) $P^*(\alpha) \leqslant s \leftrightarrow P^*(\neg\alpha) \geqslant 1 - s$
   (c) $(P^*(\alpha) \geqslant r \wedge P^*(\beta) \geqslant s \wedge P^*(\neg\alpha \vee \neg\beta) \geqslant 1) \rightarrow P^*(\alpha \vee \beta) \geqslant \min(1, r + s)$
   (d) $(P^*(\alpha) \leqslant r \wedge P^*(\beta) < s) \rightarrow P^*(\alpha \vee \beta) < r + s,\ r + s \leqslant 1$

и правила извођења:

1. Из Φ и $\Phi \rightarrow \Psi$ се изводи Ψ
2. Из α се изводи $P^*(\alpha) \geqslant 1$,
3. За тежински терм t, из $\beta \rightarrow t \geqslant s - \frac{1}{k}$ се изводи $\beta \rightarrow t \geqslant s$, за свако $k \geqslant \frac{1}{s}$ и $s > 0$

Дефиниција 2.2.4. *Формула Φ је **изводива** из скупа формула T (у ознаци $T \vdash_{Ax_{LPP_2^{ext}}} \Phi$) ако постоји највише пребројив низ формула Φ_0, Φ_1,..., Φ, тако да је свако Φ_i аксиома или формула из скупа T или је изведена из претходних формула правилом извођења. **Доказ** за формулу Φ из скупа формула T је одговарајући низ формула. Формула Φ је **теорема** (у ознаци $\vdash_{Ax^{ext}_{LPP_2}} \Phi$) ако је изводива из празног скупа.*

Дефиниција 2.2.5. *Скуп формула T је* ***конзистентан*** *ако постоји бар једна формула из For_C и бар једна формула из For_W које нису изводиве из T, у супротном је T* ***неконзистентно****. Конзистентан скуп формула T је* ***максимално конзистентан*** *ако важи:*

- *за свако $\alpha \in For_C$, ако $T \vdash_{Ax^{ext}_{LPP_2}} \alpha$, тада $\alpha \in T$ и $P^*(\alpha) \geqslant 1 \in T$ и*
- *за свако $A \in For_W$ или $A \in T$ или $\neg A \in T$.*

Скуп формула T је ***дедуктивно затворен*** *ако за свако $\Phi \in For^{ext}_{LPP_2}$, ако $T \vdash_{Ax^{ext}_{LPP_2}} \Phi$ тада $\Phi \in T$.*

2.2.4 Коректност и потпуност

За дати аксиоматски ситем доказане су теореме о коректности и јакој потпуности.

Теорема 2.2.1 (Коректност)**.** *Аксиоматски систем $Ax_{LPP_2^{ext}}$ је коректан у односу на класу $LPP^{ext}_{2,Meas}$-модела.*

Доказ теореме дат је у раду [112].

Теорема 2.2.2 (Јака потпуност)**.** *Скуп формула T је конзистентан ако и само ако T има $LPP^{ext}_{2,Meas}$-модел.*

Доказ теореме дат је у раду [112].

Проблем одлучивости и комплексности ове логике је разматран у раду [23], при чему је показано да проблем одлучивости спада у класу NP-комплетних проблема.

2.3 Вероватносне логике са приближним условним вероватноћама $LPCP$

У овом поглављу биће описана синтакса и семантика вероватносне логике са приближним условним вероватноћама. Поред тога, биће дата аксиоматизација и теореме о потпуности и одлучивости добијене логике. Посматрана логика биће означена са $LPCP$ и први пут је уведена од стране аутора Рашковића, Марковића и Огњановића у раду [101].

2.3.1 Синтакса

Нека је S јединични интервал Хардијевог поља $\mathbb{Q}[\varepsilon]$. $\mathbb{Q}[\varepsilon]$ је рекурзивно неархимедово поље које садржи све рационалне фунције од фиксиране позитивне инфинитезимале ε која припада нестандардном елементарном проширењу ${}^*\mathbb{R}$ стандардних реалних бројева ([54, 103]). Елемент ε из ${}^*\mathbb{R}$ је инфинитезимала ако $|\varepsilon| < \frac{1}{n}$ за сваки природни број n. Неки примери инфинитезимала су (у растућем поретку, ако је $\varepsilon > 0$): $\varepsilon^3 + \varepsilon^4$, $\varepsilon^2 - 5\varepsilon^6$, $\frac{\varepsilon}{100}$, 85ε, или негативне инфинитезимале: $-\varepsilon$, $-\varepsilon^2$, ... Поље $\mathbb{Q}[\varepsilon]$ садржи све рационалне бројеве. Са $\mathbb{Q}[0,1]$ биће означен скуп рационалних бројева из интервала $[0,1]$.

Језик логике са приближним условним вероватноћама се састоји од:

- пребројивог скупа исказних слова $\mathrm{Var} = \{p, q, r, \ldots\}$,
- класичних оператора $\neg$ и $\wedge$ и
- бинарних вероватносних оператора $(CP_{\leqslant s})_{s\in S}$, $(CP_{\geqslant s})_{s\in S}$ и $(CP_{\approx r})_{r\in\mathbb{Q}[0,1]}$.

Скуп For_C класичних исказних формула се дефинише на уобичајен начин. Елементи скупа For_C ће бити означавани са α, β,...

Скуп For_P^S вероватносних исказних формула је најмањи скуп Y који садржи све формуле облика:

- $CP_{\geqslant s}(\alpha, \beta)$ за $\alpha, \beta \in For_C$, $s \in S$,
- $CP_{\leqslant s}(\alpha, \beta)$ за $\alpha, \beta \in For_C$, $s \in S$ и
- $CP_{\approx r}(\alpha, \beta)$ за $\alpha, \beta \in For_C$, $r \in \mathbb{Q}[0,1]$,

и затворен је за следећа правила:

- ако A припада Y, тада је $\neg A$ у Y, и
- ако A и B припадају Y, тада је $(A \wedge B)$ у Y.

Формуле из скупа For_P^S ће бити означаване словима $A, B, \ldots$ Нека је $For_{LPCP} = For_C \cup For_P^S$. Слова $\Phi, \Psi, \ldots$ се користе за означавање формула из скупа For_{LPCP}. Битно је напоменути да мешање исказних и вероватносних формула није дозвољено, као ни угњеждавање вероватносних оператора. На пример, $\alpha \wedge CP_{\geqslant s}(\alpha, \beta)$ и $CP_{\leqslant s}(\alpha, CP_{\geqslant r}(\beta, \gamma))$ нису коректне формуле у посматраној логици, док су $CP_{>0.5+\varepsilon}(p \wedge q \wedge r, p \vee r) \wedge CP_{\leqslant 0.8-\varepsilon^2}(\neg p \wedge r, \neg p)$ и $\neg CP_{\approx 0.75}((p \vee q) \rightarrow r, \neg p \wedge \neg q)$ примери добро конструисаних формула.

Остали класични оператори ($\vee$, $\rightarrow$, $\leftrightarrow$) се дефинишу на уобичајен начин. За $\alpha \in For_C$ $A \in For_P^S$, изрази $\neg\alpha \wedge \alpha$ и $\neg A \wedge A$ биће замењени са $\bot$, док ће изрази $\neg\alpha \vee \alpha$ и $\neg A \vee A$ бити замењени са $\top$. У наставку рада израз $\pm A$ ће означавати A или $\neg A$, док се други вероватносни оператори уводе на следећи начин:

- $CP_{<s}(\alpha,\beta) =_{def} \neg CP_{\geqslant s}(\alpha,\beta)$ за $\alpha,\beta \in For_C$, $s \in S$,
- $CP_{>s}(\alpha,\beta) =_{def} \neg CP_{\leqslant s}(\alpha,\beta)$ за $\alpha,\beta \in For_C$, $s \in S$,
- $CP_{=s}(\alpha,\beta) =_{def} CP_{\geqslant s}(\alpha,\beta) \wedge CP_{\leqslant s}(\alpha,\beta)$ за $\alpha,\beta \in For_C$, $s \in S$ и
- $P_{\rho s} =_{def} CP_{\rho s}(\alpha,\top)$ за $\alpha \in For_C$ и $\rho \in \{\geqslant,\leqslant,>,<,=,\approx\}$.

Оператор $P_{\rho s}$ је вероватносни оператор уведен у логици LPP_2, која се сада може посматрати као подлогика логике $LPCP$.

2.3.2 Семантика

Семантика за For_{LPCP} се, такође, заснива на Крипкеовим моделима [101].

Дефиниција 2.3.1. *$LPCP$-модел је структура $\mathbf{M} = \langle W, H, \mu, v\rangle$ где је:*

- *W непразан скуп објеката званих светови,*
- *H алгебра подскупова од W, тј. H садржи W и затворен је за комплемент и коначну унију,*
- *$\mu : H \to S$ коначна адитивна вероватносна мера где:*
  - *за свако $X \in H$, $\mu(X) \geq 0$,*
  - *$\mu(W) = 1$, и*
  - *$\mu(X_1 \cup X_2) = \mu(X_1) + \mu(X_2)$, за све дисјунктне скупове X_1, $X_2 \in H$,*
- *$v : W \times \mathrm{Var} \to \{\mathrm{true}, \mathrm{false}\}$ функција која сваком свету $w \in W$ додељује истинитосну валуацију $v(w)$ за исказна слова.*

Функција v је проширена до истинитосне валуације за све исказне формуле на уобичајен начин. Нека је $\mathbf{M}$ $LPCP$-модел и $\alpha \in For_C$. Скуп $\{w : v(w)(\alpha) = \mathrm{true}\}$ је означен са $[\alpha]_{\mathbf{M}}$.

Дефиниција 2.3.2. *$LPCP$-модел $\mathbf{M}$ је **мерљив** ако $[\alpha]_{\mathbf{M}} \in H$ за свако $\alpha \in For_C$. $LPCP$-модел $\mathbf{M}$ је **уредан** ако само празни скупови имају вероватноћу нула. $LPCP_{Meas,Neat}$ означава класу свих уредних и мерљивих $LPCP$-модела.*

Услов да модел буде уредан је уведен да би посматран модел био подкласа од ${}^*\mathbb{R}$-вероватносних модела из [59, 63]. Ово ће олакшати наредна објашњења за примену овог система на дифолтно резоновање, које ће бити детаљно објашњено у глави 3. Сви резултати се могу доказати и за класу мерљивих $LPCP$-модела.

Дефиниција 2.3.3. ***Релација задовољивости*** $\models \subset LPCP_{Meas,Neat} \times For_{LPCP}$ *је дефинисана следећим условима за сваки* $LPCP_{Meas,Neat}$*-модел* $\mathbf{M}$*:*

1. *ако је* $\alpha \in For_C$, $\mathbf{M} \models \alpha$ *ако* $(\forall w \in W)v(w)(\alpha) = \text{true}$,
2. $\mathbf{M} \models CP_{\leq s}(\alpha, \beta)$ *ако је или* $\mu([\beta]_{\mathbf{M}}) = 0$ *и* $s = 1$ *или* $\mu([\beta]_{\mathbf{M}}) > 0$ *и* $\frac{\mu([\alpha \wedge \beta]_{\mathbf{M}})}{\mu([\beta]_{\mathbf{M}})} \leq s$,
3. $\mathbf{M} \models CP_{\geq s}(\alpha, \beta)$ *ако је или* $\mu([\beta]_{\mathbf{M}}) = 0$ *или* $\mu([\beta]_{\mathbf{M}}) > 0$ *и* $\frac{\mu([\alpha \wedge \beta]_{\mathbf{M}})}{\mu([\beta]_{\mathbf{M}})} \geq s$,
4. $\mathbf{M} \models CP_{\approx r}(\alpha, \beta)$ *ако је или* $\mu([\beta]_{\mathbf{M}}) = 0$ *и* $r = 1$ *или* $\mu([\beta]_{\mathbf{M}}) > 0$ *и за сваки позитиван цео број* n, $\frac{\mu([\alpha \wedge \beta]_{\mathbf{M}})}{\mu([\beta]_{\mathbf{M}})} \in [\max\{0, r - 1/n\}, \min\{1, r + 1/n\}]$.
5. *ако је* $A \in For_P^S$, $\mathbf{M} \models \neg A$ *ако* $\mathbf{M} \not\models A$,
6. *ако су* $A, B \in For_P^S$, $\mathbf{M} \models A \wedge B$ *ако* $\mathbf{M} \models A$ *и* $\mathbf{M} \models B$.

Услов 3 је формулисан на основу претпоставке да је условна вероватноћа 1, кад год је вероватноћа услова 0. Услов 4 је еквивалентан запису да је условна вероватноћа једнака $r - \varepsilon_i$ (или $r + \varepsilon_i$) за неку инфинитезималу $\varepsilon_i \in S$. Лако се види да се дефинисани оператори понашају као што је очекивано, на пример, $\mathbf{M} \models P_{<s}\alpha$ ако и само ако $\mu([\alpha]_{\mathbf{M}}) < s$.

Дефиниција 2.3.4. *Формула* $\Phi \in For_{LPCP}$ *је* ***задовољива*** *ако постоји* $LPCP_{Meas,Neat}$*-модел* $\mathbf{M}$ *такав да је* $\mathbf{M} \models \varphi$*;* Φ *је* ***ваљана*** *ако за сваки* $LPCP_{Meas,Neat}$*-модел* $\mathbf{M}$, $\mathbf{M} \models \Phi$*; скуп формула је* ***задовољив*** *ако постоји* $LPCP_{Meas,Neat}$*-модел* $\mathbf{M}$ *у коме је свака формула скупа задовољива.*

2.3.3 Аксиоматски систем

Аксиоматски систем Ax_{LPCP} за логику $LPCP$ садржи следеће схеме аксиома [101]:

1. Све For_C-инстанце исказних таутологија
2. Све For_P^S-инстанце исказних таутологија
3. $CP_{\geq 0}(\alpha, \beta)$
4. $CP_{\leq s}(\alpha, \beta) \rightarrow CP_{<t}(\alpha, \beta)$, $t > s$
5. $CP_{<s}(\alpha, \beta) \rightarrow CP_{\leq s}(\alpha, \beta)$
6. $P_{\geq 1}(\alpha \leftrightarrow \beta) \rightarrow (P_{=s}\alpha \rightarrow P_{=s}\beta)$
7. $P_{\leq s}\alpha \leftrightarrow P_{\geq 1-s}\neg\alpha$

8. $(P_{=s}\alpha \wedge P_{=t}\beta \wedge P_{\geq 1}\neg(\alpha \wedge \beta)) \rightarrow P_{=\min(1,s+t)}(\alpha \vee \beta)$
9. $P_{=0}\beta \rightarrow CP_{=1}(\alpha, \beta)$
10. $(P_{=t}\beta \wedge P_{=s}(\alpha \wedge \beta)) \rightarrow CP_{=s/t}(\alpha, \beta)$, $t \neq 0$
11. $CP_{\approx r}(\alpha, \beta) \rightarrow CP_{\geq r_1}(\alpha, \beta)$, за све рационалне $r_1 \in [0, r)$
12. $CP_{\approx r}(\alpha, \beta) \rightarrow CP_{\leq r_1}(\alpha, \beta)$, за све рационалне $r_1 \in (r, 1]$

и правила извођења:

1. Из φ и $\varphi \rightarrow \psi$ се изводи ψ.
2. Ако је $\alpha \in For_C$, из α се изводи $P_{\geq 1}\alpha$.
3. Из $A \rightarrow P_{\neq s}\alpha$, за свако $s \in S$, се изводи $A \rightarrow \bot$.
4. За свако $r \in \mathbb{Q}[0,1]$, из $A \rightarrow CP_{\geq r-1/n}(\alpha, \beta)$, за сваки цео број $n \geq 1/r$, и $A \rightarrow CP_{\leq r+1/n}(\alpha, \beta)$ за сваки цео број $n \geq 1/(1-r)$, се изводи $A \rightarrow CP_{\approx r}(\alpha, \beta)$.

Ако се за β стави $\top$, аксиома 3 тврди да је свака формула задовољива у скупу светова чија је вероватноћа најмање 0. Аксиома 6 значи да еквивалентне формуле имају исту вероватноћу. Аксиома 8 обезбеђује коначну адитивност вероватноће, тј. ако су скупови светова у којима су α и β задовољене дисјунктни, тада је вероватноћа скупа светова у којима је $\alpha \vee \beta$ задовољена сума вероватноћа претходна два скупа. Аксиома 9 је додата да би се потврдило да је условна вероватноћа 1, кад год је вероватноћа услова 0. Аксиома 10 изражава стандардну дефиницију условне вероватноће. Аксиоме 11 и 12 и правило извођења 4 описују везу између стандардне условне вероватноће и условне вероватноће инфинитезимално блиске неком рационалном броју $r \in \mathbb{Q}[0,1]$. На основу аксиома 3 и 7, и правила извођења 2 добија се правило извођења: из α изводи се $P_{=1}\alpha$. Правила извођења 3 и 4 су бесконачна правила. Правило 3 гарантује да вероватноћа формуле припада скупу S.

Дефиниција 2.3.5. [101] *Формула* Φ *је* ***синтаксна последица*** *скупа формула* T *(у ознаци* $T \vdash_{Ax_{LPCP}} \Phi$*) ако постоји највише пребројив низ формула* $\Phi_0, \Phi_1, \ldots, \Phi$*, таквих да је свака формула* Φ_i *аксиома или формула из скупа* T*, или је помоћу неког правила извођења изведена из претходних формула. Низ формула* $\Phi_0, \Phi_1, \ldots, \Phi$ *је* ***доказ*** *за* Φ *из скупа* T*.*

Дужина доказа је пребројиви ординал. Сличан приступ је дат у [49].

Дефиниција 2.3.6. [101] *Формула* Φ *је* ***теорема*** *(*$\vdash_{Ax_{LPCP}} \Phi$*) ако је синтаксна последица празног скупа формула. Скуп формула* T *је* ***конзистентан*** *ако постоји бар једна формула из* For_C *и бар једна формула из* For_P^S *које нису синтаксне последице скупа* T*.*

За логику $LPCP$ је у раду [101] дат доказ теореме о јакој потпуности.

Теорема 2.3.1 (Јака потпуност)**.** *Скуп формула* T *је конзистентан ако и само ако* T *има* $LPCP_{Meas,Neat}$*-модел.*

2.3.4 Одлучивост

Пошто је проблем задовољивости формула у исказној логици одлучив, да би се доказала одлучивост логике $LPCP$, довољно је показати да је проблем задовољивости вероватносних формула одлучив. Најпре ће бити описан поступак којим се проблем задовољивости вероватносних формула своди на проблем линеарног програмирања, а затим ће то бити искоришћено у доказу теореме.

Нека је A вероватносна формула и нека су $p_1, \ldots, p_n$ сва исказна слова која се јављају у A. Атом a за A је формула $\pm p_1 \wedge \ldots \wedge \pm p_n$. За различите атоме a_i и a_j важи $\vdash a_i \rightarrow \neg a_j$. Са $\mathrm{At}(A)$ ће бити означен скуп свих атома из A. Очигледно је да је $|\mathrm{At}(A)| = 2^n$.

На основу исказног резоновања се лако показује да је свака формула A еквивалентна са формулом: $DNF(A) = \bigvee_{i=1}^{m} \bigwedge_{j=1}^{k_i} \pm X_{i,j}(p_1, \ldots, p_n)$ која представља дисјунктивну нормалну форму за формулу A при чему је:

- $X_{i,j} \in \{CP_{\geqslant s}, CP_{\leqslant s}\}_{s \in S} \cup \{CP_{\approx r}\}_{r \in \mathbb{Q}[0,1]}$,
- $X_{i,j}(p_1, \ldots, p_n)$ означава да је вероватносна формула на коју делује оператор $X_{i,j}$ у канонској дисјунктивној нормалној форми, тј. исказна формула је дисјункција атома из A.

Очигледно је да, да би се показала одлучивост логике $LPCP$, довољно је показати да је проблем задовољивости вероватносне формуле облика $\bigwedge_{j=1}^{k_i} \pm X_{i,j}(p_1, \ldots, p_n)$ одлучив.

За сваку формулу условне вероватноће ($\pm CP_{\geqslant s}(\alpha, \beta)$, $\pm CP_{\leqslant s}(\alpha, \beta)$ и $\pm CP_{\approx r}(\alpha, \beta)$) се разликују два случаја:

1. вероватноћа формуле β је нула, у ком случају
   - $CP_{\geqslant s}(\alpha, \beta)$, за $s \in S$, $\neg CP_{\leqslant s}(\alpha, \beta)$, за $s \in S \setminus \{1\}$, $CP_{\leqslant 1}(\alpha, \beta)$ и $CP_{\approx 1}(\alpha, \beta)$ важе и могу се избрисати из формуле, док
   - $\neg CP_{\geqslant s}(\alpha, \beta)$, за $s \in S$, $CP_{\leqslant s}(\alpha, \beta)$, $CP_{\approx r}(\alpha, \beta)$, за $s \in S \setminus \{1\}$ $r \in \mathbb{Q}[0,1] \setminus \{1\}$, $\neg CP_{\leqslant 1}(\alpha, \beta)$ $\neg CP_{\approx 1}(\alpha, \beta)$ не важе, па цела конјункција није задовољива
2. вероватноћа формуле β је већа од нуле.

Ово за последицу има да је довољно доказати одлучивост проблема задовољивости формула које су конјункције формула са условним вероватноћама облика: $\pm CP_{\geqslant s}(\alpha, \beta)$, $\pm CP_{\leqslant s}(\alpha, \beta)$ и $\pm CP_{\approx r}(\alpha, \beta)$, таквих да је вероватноћа β већа од 0.

У наредном кораку се проблем задовољивости своди на проблем линеарног програмирања. Исти приступ за доказ одлучивости је коришћен и у радовима [23, 100, 88] који се односе на стандардне реално-вредносне вероватноће. Међутим, у логици $LPCP$ кодомен за вероватноћу је рекурзиван и садржи нестандардне вредности и,

додатно, оператор облика $CP_{\approx r}$ се не јавља у поменутим радовима. Ово значи да редукција мора да се обави пажљиво да би добијени систем могао да се искористи за утврђивање одлучивости, што би у овом случају значило да би могла да се примени Фурије-Моцкин метода елиминације (FME). Идеја је да се елиминишу симболи $\approx$ и $\not\approx$ и да се ситем реши у проширењу $\mathbb{Q}[\varepsilon]$.

У наставку ће бити коришћене следеће ознаке:

- x_i означава меру атома $a_i \in At(A)$, $i = 1, \ldots, 2^n$,
- $a_i \models \alpha$ означава да се атом a_i појављује у канонској дисјунктивној нормалној форми исказне формуле α,
- $\sum(\alpha)$ означава $\sum_{a_i \in At(A): a_i \models \alpha} x_i$, и
- $\mathrm{C}\sum(\alpha, \beta)$ означава $\frac{\sum(\alpha \wedge \beta)}{\sum(\beta)}$.

$[\alpha]$ означава скуп атома који задовољавају α. Пошто је $[\alpha] = \cup_{a_i \in At(A): a_i \models \alpha}[a_i]$, различити атоми се међусобно искључују (тј., $[a_i] \cap [a_j] = \emptyset$ за $i \neq j$).

У наставку ће бити разматрана формула A облика:

$$\begin{array}{c}(\wedge_{i=1,I} \pm CP_{\geqslant s_i}(\alpha_i, \beta_i)) \wedge (\wedge_{j=1,J} \pm CP_{\leqslant s_j}(\alpha_j, \beta_j)) \\ \wedge(\wedge_{k=1,K} \pm CP_{\approx r_k}(\alpha_k, \beta_k)).\end{array} \tag{2.3.1}$$

Формула A је задовољива ако и само ако је задовољив следећи систем:

$\sum_{i=1}^{2^n} x_i = 1$	
$x_i \geqslant 0$	за $i = 1, 2^n$
$\sum(\beta) > 0$	за сваку формулу β која се појављује у формулама облика $\pm CP_\diamond(\alpha, \beta)$ из A, и $\diamond \in \{\geqslant s_i, \leqslant s_j, \approx r_k\}$
$\mathrm{C}\sum(\alpha_i, \beta_i) \geqslant s_i$	за сваку формулу $CP_{\geqslant s_i}(\alpha_i, \beta_i)$ из A
$\mathrm{C}\sum(\alpha_i, \beta_i) < s_i$	за сваку формулу $\neg CP_{\geqslant s_i}(\alpha_i, \beta_i)$ из A
$\mathrm{C}\sum(\alpha_j, \beta_j) \leqslant s_j$	за сваку формулу $CP_{\leqslant s_j}(\alpha_j, \beta_j)$ из A
$\mathrm{C}\sum(\alpha_j, \beta_j) > s_j$	за сваку формулу $\neg CP_{\leqslant s_j}(\alpha_j, \beta_j)$ из A
$\mathrm{C}\sum(\alpha_k, \beta_k) \approx r_k$	за сваку формулу $CP_{\approx r_k}(\alpha_k, \beta_k)$ из A
$\mathrm{C}\sum(\alpha_k, \beta_k) \not\approx r_k$	за сваку формулу $\neg CP_{\approx r_k}(\alpha_k, \beta_k)$ из A.

Прве две врсте система се односе на особине вероватноће, тј. сума вероватноћа свих атома мора бити 1, док вероватноћа сваког атома мора бити ненегативан број.

Дати систем се може даље поједноставити тако што ће сваки израз облика $\mathrm{C}\sum(\alpha_k, \beta_k) \approx r_k$ бити записан као:

$$\mathrm{C}\sum(\alpha_k, \beta_k) - r_k \approx 0 \quad \text{и} \quad \mathrm{C}\sum(\alpha_k, \beta_k) - r_k \geqslant 0 \tag{2.3.2}$$

или

$$\mathrm{C}\sum(\alpha_k, \beta_k) - r_k \approx 0 \quad \text{и} \quad r_k - \mathrm{C}\sum(\alpha_k, \beta_k) \geqslant 0. \tag{2.3.3}$$

Израз (2.3.2) одговара изразу $\mathrm{C}\sum(\alpha_k, \beta_k) \geqslant r_k$ док израз (2.3.3) одговара изразу $r_k \geqslant \mathrm{C}\sum(\alpha_k, \beta_k)$.

Слично, сваки израз облика $\mathrm{C}\sum(\alpha_k, \beta_k) \not\approx r_k$ се може записати као

$$\mathrm{C}\sum(\alpha_k, \beta_k) - r_k \not\approx 0 \quad \text{и} \quad \mathrm{C}\sum(\alpha_k, \beta_k) - r_k > 0 \tag{2.3.4}$$

или

$$\mathrm{C}\sum(\alpha_k, \beta_k) - r_k \not\approx 0 \quad \text{и} \quad r_k - \mathrm{C}\sum(\alpha_k, \beta_k) > 0. \tag{2.3.5}$$

Даље се разматра систем који садржи изразе облика (2.3.2 - 2.3.5) уместо $\mathrm{C}\sum(\alpha_k, \beta_k) \approx r_k$, и $\mathrm{C}\sum(\alpha_k, \beta_k) \not\approx r_k$, респективно. За систем овог облика биће коришћена ознака $S(\overrightarrow{x}, \varepsilon)$. Даље је

$$\mathrm{C}\sum(\alpha_k, \beta_k) - r_k \approx 0 \quad \text{и} \quad \mathrm{C}\sum(\alpha_k, \beta_k) - r_k \geqslant 0 \tag{2.3.6}$$
$$\text{еквивалентно са} \quad (\exists n_k \in \mathbb{N})\ 0 \leqslant \mathrm{C}\sum(\alpha_k, \beta_k) - r_k < n_k \cdot \varepsilon,$$

$$\mathrm{C}\sum(\alpha_k, \beta_k) - r_k \approx 0 \quad \text{и} \quad r_k - \mathrm{C}\sum(\alpha_k, \beta_k) \geqslant 0 \tag{2.3.7}$$
$$\text{еквивалентно са} \quad (\exists n_k \in \mathbb{N})\ 0 \geqslant \mathrm{C}\sum(\alpha_k, \beta_k) - r_k > -n_k \cdot \varepsilon,$$

$$\mathrm{C}\sum(\alpha_k, \beta_k) - r_k \not\approx 0 \quad \text{и} \quad \mathrm{C}\sum(\alpha_k, \beta_k) - r_k > 0 \tag{2.3.8}$$
$$\text{еквивалентно са} \quad (\exists n_k \in \mathbb{N})\mathrm{C}\sum(\alpha_k, \beta_k) - r_k > \frac{1}{n_k},$$

$$\mathrm{C}\sum(\alpha_k, \beta_k) - r_k \not\approx 0 \quad \text{и} \quad r_k - \mathrm{C}\sum(\alpha_k, \beta_k) > 0 \tag{2.3.9}$$
$$\text{еквивалентно са} \quad (\exists n_k \in \mathbb{N})\mathrm{C}\sum(\alpha_k, \beta_k) - r_k < -\frac{1}{n_k}.$$

Пошто у добијеном ситему има коначно много израза облика (2.3.2 – 2.3.5), може се користити јединствено $n_0 \in \mathbb{N}$ уместо више n_k's у изразима (2.3.6 – 2.3.9). Ознака $S(\overrightarrow{x}, n_0, \varepsilon)$ ће бити коришћена за означавање добијеног система. Тада,

$$S(\overrightarrow{x}, \varepsilon) \text{ има решење у } \mathbb{Q}[\varepsilon] \text{ акко } S(\overrightarrow{x}, n_0, \varepsilon) \text{ има решење у } \mathbb{Q}[\varepsilon], \tag{2.3.10}$$

и, пошто је $\mathbb{Q}[\varepsilon]$ густо у ${}^*\mathbb{R}$, за свако фиксирано и коначно n_0,

$$S(\overrightarrow{x}, n_0, \varepsilon) \text{ има решење у } \mathbb{Q}[\varepsilon] \text{ акко } S(\overrightarrow{x}, n_0, \varepsilon) \text{ има решење у } {}^*\mathbb{R}. \tag{2.3.11}$$

Параметар n_0 није фиксиран у изразима 2.3.10 и 2.3.11, па се може заменити новим, бесконачним, али фиксираним, параметром K. Улога параметра K је да помогне да се избегне стандардан приступ у анализи неједнакости, где би се врло често користили аргументи у облику „важи за све довољно велике целе бројеве". Пошто је K позитиван бесконачан цео број, ако неједнакост важи за свако n веће од неког фиксног n_0, на основу принципа преливања важиће и за K. У другом смеру, као последица принципа подливања, ако неједнакост важи за сваки бесконачан број мањи од K, важиће и за неки коначан позитиван цео број. Биће разматран скуп

$$O = \{n \in {}^*\mathbb{N} : S(\overrightarrow{x}, n, \varepsilon) \text{ има решење у } {}^*\mathbb{R}\}$$

O је скуп који садржи све природне бројеве веће од фиксираног природног броја n'. Користећи принципе преливања и подливања, може се закључити да ако је $S(\overrightarrow{x}, \varepsilon)$ решиво у $\mathbb{Q}[\varepsilon]$, тада O садржи све бесконачно велике бројеве из ${}^*\mathbb{N}$ који су мањи од фиксираног бесконачно великог броја K, тј. за неко $n' \in \mathbb{N}$ и $K \in {}^*\mathbb{N} \setminus \mathbb{N}$ важи $[n', K] = \{n \in {}^*\mathbb{N} : n' \leqslant n \leqslant K\} \subset O$. Тада

$$S(\overrightarrow{x}, n_0, \varepsilon) \text{ има решење у } {}^*\mathbb{R} \text{ акко } S(\overrightarrow{x}, K, \varepsilon) \text{ има решење у } {}^*\mathbb{R}. \tag{2.3.12}$$

Параметар K се може изабрати тако да за свако $k \in \mathbb{N}$ важи $K^k \cdot \varepsilon \approx 0$. На овај начин се систем решава у уређеном пољу рационалних израза са рационалним коефицијентима који зависе од ε и K. Сабирање и множење у овом пољу се обавља на уобичајени начин. Уређење поља је одређено уређењем скупа полинома који зависе од ε и K. Сваки полином у $\mathbb{Q}(\varepsilon, K)$ се може записати у облику $Q_0(K)\varepsilon^0 + Q_1(K)\varepsilon^1 + \ldots + Q_n(K)\varepsilon^n$, где су $Q_i(K)$ полиноми по K са рационалним коефицијентима. Поређење полинома $Q_1(\varepsilon, K)$ и $Q_2(\varepsilon, K)$ се обавља на следећи начин:

$$\begin{aligned}
& Q_1(\varepsilon, K) < Q_2(\varepsilon, K) \\
\text{акко}\quad & Q_{1,0}(K)\varepsilon^0 + Q_{1,1}(K)\varepsilon^1 + \ldots + Q_{1,n_1}(K)\varepsilon^{n_1} \\
& \qquad < Q_{2,0}(K)\varepsilon^0 + Q_{2,1}(K)\varepsilon^1 + \ldots + Q_{2,n_2}(K)\varepsilon^{n_2} \\
\text{акко}\quad & Q_{1,i}(K) < Q_{2,i}(K) \text{ за неко } i \leq \max\{n_1, n_2\} \text{ и } Q_{1,j}(K) = Q_{2,j}(K) \\
& \text{за } j < i \quad (Q_{1,i}(K) = 0 \text{ за } i > n_1 \text{ и } Q_{2,i}(K) = 0 \text{ за } i > n_2)
\end{aligned}$$

и

$$\begin{aligned}
& Q_{1,i}(K) < Q_{2,i}(K) \\
\text{акко}\quad & q_{1,i,m_1}K^{m_1} + \cdots + q_{1,i,1}K + q_{1,i,0} < q_{1,i,m_2}K^{m_2} + \cdots + q_{2,i,1}K + q_{2,i,0} \\
\text{акко}\quad & q_{1,i,r} < q_{2,i,r} \text{ за неко } r \leq \max\{m_1, m_2\} \text{ и } q_{1,i,t} = q_{2,i,t} \text{ за } t < r \\
& (q_{1,i,r} = 0 \text{ за } r > m_1 \text{ и } q_{2,i,r} = 0 \text{ за } r > m_2)
\end{aligned}$$

На пример,

$$\left(\frac{1}{2}K^2+K\right)+\quad\left(\frac{1}{3}K^3+\frac{1}{2}K^2+K\right)\varepsilon+\varepsilon^2<$$
$$<\left(\frac{1}{2}K^2+K\right)+\left(\frac{1}{3}K^3+\frac{2}{3}K^2+2K+1\right)\varepsilon$$

пошто је $\frac{1}{2}K^2+K=\frac{1}{2}K^2+K$ ($\frac{1}{2}=\frac{1}{2}$, $1=1$) и $\frac{1}{3}K^3+\frac{1}{2}K^2+K<\frac{1}{3}K^3+\frac{2}{3}K^2+2K+1$ ($\frac{1}{3}=\frac{1}{3}$, $\frac{1}{2}<\frac{2}{3}$).

Ако је $Q_1'(\varepsilon,K)>0$ и $Q_2'(\varepsilon,K)>0$, тада је

$$\frac{Q_1(\varepsilon,K)}{Q_1'(\varepsilon,K)}<\frac{Q_2(\varepsilon,K)}{Q_2'(\varepsilon,K)}\text{ акко }Q_1(\varepsilon,K)\cdot Q_2'(\varepsilon,K)<Q_2(\varepsilon,K)\cdot Q_1'(\varepsilon,K).$$

На основу овога, дозвољено је (не)једнакости множити делиоцима у изразима $\mathrm{C}\sum(\alpha,\beta)$, чиме се добија систем линеарних (не)једнакости облика

$$\sum(\alpha\wedge\beta)-s\sum(\beta)\quad\rho\quad 0, \tag{2.3.13}$$

где је s рационална функција која зависи од K и ε и $\rho\in\{\geqslant,>,<,\leqslant\}$. За решавање овог система може се применити Фурије-Моцкин процедура елиминације, која модификује систем тако да се у новодобијеном систему не појављује непозната x_i, а системи су еквивалентни[2]. Током примене процедуре коефицијнти у систему остају рационални изрази по ϵ и K. Када се елиминишу све непознате, преостаје да се потврди задовољивост релација међу полиномима по ϵ и K.

На основу свега претходног следи да је проблем испитивања да ли $S(\overrightarrow{x},K,\varepsilon)$ има решење у ${}^*\mathbb{R}$ одлучив. На основу еквиваленција (2.3.10) – (2.3.12) следи да је проблем испитивања да ли $S(\overrightarrow{x},\varepsilon)$ има решење у $\mathbb{Q}[\varepsilon]$, такође, одлучив.

Ако је $S(\overrightarrow{x},\varepsilon)$ решиво, тада се може дефинисати $LPCP$-модел $\mathbf{M}=\langle W,H,\mu,v\rangle$ такав да је $W=At(A)$, $H=2^W$, μ је дефинисано решењем система $S(\overrightarrow{x},\varepsilon)$ и v задовољава $v(a)(p)=\text{true}$ акко се p, али не и $\neg p$ јавља у атому a. Очигледно је да је $\mathbf{M}\models A$. Међутим, иако $S(\overrightarrow{x},\varepsilon)$ има решење, неки x_i могу имати вредност 0. То значи да $\mathbf{M}$ не задовољава услов уредности, тј. неки непразни скупови светова (представљени одговарајућим атомима који важе у тим световима) имају вероватноћу нула. У овом случају, ови светови се могу уклонити из модела и новодобијени модел се означава са $\mathbf{M}'$. Лако се уочава да за свако $A\in For_P^S$ важи $\mathbf{M}\models A$ акко $\mathbf{M}'\models A$.

Теорема 2.3.2. *Проблем $LPCP^S_{Meas,Neat}$-задовољивости је одлучив.*

Пример 2.3.1. *Нека је дата формула $A=CP_{\geqslant 0.36+3\varepsilon}(p\vee q,\neg p\vee q)\wedge CP_{\approx 0.5}(p,p\vee q)\wedge CP_{<0.9-0.3\varepsilon^2}(p,q)$. Скуп атома формуле A чине атоми $a_1=p\wedge q$, $a_2=p\wedge\neg q$,*

[2]Детаљан опис процедуре биће дат у поглављу А.1.

$a_3 = \neg p \wedge q$, $a_4 = \neg p \wedge \neg g$. Нека x_i означава меру атома a_i. Исказне формуле у формули A су $p \vee q$, $\neg p \vee q$, p и q, док су скупови атома који их задовољавају следећи:

α	$p \vee q$	$\neg p \vee q$	p	q
$[\alpha]$	a_1, a_2, a_3	a_1, a_3, a_4	a_1, a_2	a_1, a_3

Тада је A задовољиво ако и само ако је следећи систем задовољив:

$$
\begin{array}{l}
x_1 + x_2 + x_3 + x_4 = 1 \\
x_i \geqslant 0, \qquad i = 1, 2, 3, 4 \\
x_1 + x_3 + x_4 > 0 \qquad x_1 + x_2 + x_3 > 0 \qquad x_1 + x_3 > 0 \\
(0.64 - 3\varepsilon)x_1 + (0.64 - 3\varepsilon)x_3 + (-0.36 - 3\varepsilon)x_4 \geqslant 0 \\
(0.5 - \varepsilon)x_1 + (0.5 - \varepsilon)x_2 + (-0.5 - \varepsilon)x_3 < 0 \\
(0.5 + \varepsilon)x_1 + (0.5 + \varepsilon)x_2 + (-0.5 + \varepsilon)x_3 > 0 \\
(0.1 + 0.3\varepsilon^2)x_1 + (-0.9 + 0.3\varepsilon^2)x_3 < 0
\end{array}
$$

За доказивање задовољивости система довољно је наћи једно решење система:

$$
\begin{array}{ll}
x_1 = 0.14 + 2\varepsilon & x_2 = 0.24 - 3.6\varepsilon + 0.6\varepsilon^2 \\
x_3 = 0.4 - 0.1\varepsilon - 0.3\varepsilon^2 & x_4 = 0.2 + 1.7\varepsilon - 0.3\varepsilon^2
\end{array}
$$

На основу датих трансформација, показано је да се проблем задовољивости своди на проблем линеарног програмирања који се може описати на следећи начин:

$$
\begin{array}{ll}
 & \min z(x) \\
s.t. & \sum_{i=1}^{2^n} c_{ij} x_i \ \rho \ 0 \quad j = 1, 2, \ldots L, \quad \rho \in \{\geqslant, >, <, \leqslant\} \\
 & \sum_{i=1}^{2^n} x_i = 1 \\
 & x_i \geqslant 0 \qquad\qquad i = 1, 2, \ldots, 2^n
\end{array}
$$

где су неједнакости $\sum_{i=1}^{2^n} c_{ij} x_i \ \rho \ 0$ добијене поједностављивањем неједнакости облика (2.3.13), L представља број ових неједенкости и све вредности $c_{ij}, x_i (i = 1, 2, \ldots, 2^n, j = 1, 2, \ldots, L)$ припадају Хардијевом пољу, тј. то су рационални изрази по ε и K. Функција циља $z(x)$ је дефинисана на следећи начин

$$z(x) = \sum_{i=1}^{L} d_i(x),$$

где је

$$d_i(x) = \begin{cases} (\sum_{i=1}^{2^n} c_{ij} x_i)^2 & \text{ако } j\text{-та неједнакост није задовољена,} \\ 0 & \text{у супротном,} \end{cases}$$

У поглављу 5 ће функција циља бити детаљније разматрана, где ће бити објашњен и хеуристички приступ у решавању овог проблема.

Глава 3

Дифолтно закључивање

У овом поглављу биће представљено резоновање у дифолтним логикама, као и веза дифолтних логика и логика са условним приближним вероватноћама. Дифолтно резоновање је облик немонотоног резоновања базираног на извођењу закључака на основу општег скупа правила која могу имати изузетке или скупа чињеница које представљају доступне информација које нису потпуне ([5]). Најпознатији пример ове врсте резоновања је: ако *"птице лете"*, *"пингвини су птице"* и *"пингвини не лете"* да ли се може закључити да *"ако је Твити птица и пингвин да ли то значи да Твити не лети"*? Од 1980. су развијени различити системи за дифолтно резоновање и детаљан преглед ових система је дат у раду [5]. Главна пажња ће бити посвећена систему **P**, чије су основе дате у раду [59].

3.1 Дифолтно резоновање у систему P

Као и код вероватносних логика, језик посматране дифолтне логике се састоји из пребројивог скупа исказних слова $\mathrm{Var} = \{p, q, r, \ldots\}$ и класичних оператора $\neg$, $\wedge$, $\vee$, $\rightarrow$. Скуп исказних формула ће, као и у претходном поглављу, бити означен са For_C и дефинисан на уобичајени начин.

Ако су α и β исказне формуле из скупа For_C, тада се пар (α, β), означен са $\alpha \rightarrowtail \beta$, назива **дифолтно правило** или, скраћено, **дифолт** (енг. *default rule*, односно *default*). Различити аутори користе различите ознаке (на пример $\rightarrow$ или $\mid\!\sim$) да означе 'дифолтну импликацију'. Да би се избегла забуна, овде је уведен нов симбол. Неке од уобичајених интуитивних интерпретација записа $\alpha \rightarrowtail \beta$ су: 'из α је разумно закључити β'; 'у општем случају, ако је α тада је β (уз могуће постојање неког изузетка)'; 'α је довољно добар разлог да се верује у β'; 'β је веродостојан закључак из α'; 'ако је α, нормално је да је β'; 'ако је α тачно, тада сам склон да закључим да је β тачно'; и тако даље.

База дифолта је скуп $\Delta = \{\alpha_i \rightarrowtail \beta_i \mid i \in I\}$ дифолтних правила. База дифолта садржи чињенице које су подложне поништењу. Дифолтно резоновање се описује

у терминима последичних релација $|\sim_{\mathbf{P}}$ које одређују скуп дифолта који су $|\sim_{\mathbf{P}}$-последица база дифолта. У радовима [59, 63] је релација $|\sim_{\mathbf{P}}$ дефинисана скупом особина, названим Систем **P**, и сматра се језгром дифолтног резоновања ($\models$ означава класичну релација ваљаности):

REF $\alpha \rightarrowtail \alpha$ (рефлексивност);

LLE ако је $\models \alpha \leftrightarrow \beta$, из $\alpha \rightarrowtail \gamma$ се изводи $\beta \rightarrowtail \gamma$ (лева логичка еквиваленција;)

RW ако је $\models \alpha \rightarrow \beta$, из $\gamma \rightarrowtail \alpha$ се изводи $\gamma \rightarrowtail \beta$ (десно ослабљење);

CUT из $\alpha \wedge \beta \rightarrowtail \gamma$ $\alpha \rightarrowtail \beta$ се изводи $\alpha \rightarrowtail \gamma$;

CM из $\alpha \rightarrowtail \beta$ и $\alpha \rightarrowtail \gamma$ се изводи $\alpha \wedge \gamma \rightarrowtail \beta$ („опрезна" монотоност);

OR из $\alpha \rightarrowtail \gamma$ и $\beta \rightarrowtail \gamma$ се изводе $\alpha \vee \beta \rightarrowtail \gamma$.

За дату базу дифолта $\Delta = \{\alpha_i \rightarrowtail \beta_i \mid i \in I\}$, ознака $\Delta|\sim_{\mathbf{P}} \alpha \rightarrowtail \beta$ означава да се $\alpha \rightarrowtail \beta$ може извести из Δ користећи систем **P**.

Пример 3.1.1. *Нека је* $\Delta = \{b \rightarrowtail f, p \rightarrowtail b, p \rightarrowtail \neg f\}$ *где* b *означава 'птице' (birds),* f *означава 'летење' (flies) и* p *означава 'пингвине' (penguin). Тада* $\Delta|\sim_{\mathbf{P}} b \rightarrowtail \neg p$.

1. $b \rightarrowtail f$	*(претпоставка)*
2. $p \rightarrowtail b$	*(претпоставка)*
3. $p \rightarrowtail \neg f$	*(претпоставка)*
4. $p \wedge b \rightarrowtail \neg f$	*CM(2,3)*
5. $p \wedge b \rightarrowtail p \rightarrow \neg f$	*RW(4),* $\models \neg f \rightarrow (p \rightarrow \neg f)$
6. $\neg p \wedge b \rightarrowtail \neg p \wedge b$	*REF*
7. $\neg p \wedge b \rightarrowtail p \rightarrow \neg f$	*RW(6),* $\models \neg p \wedge b \rightarrow (p \rightarrow \neg f)$
8. $(p \wedge b) \vee (\neg p \wedge b) \rightarrowtail p \rightarrow \neg f$	*OR(5,7)*
9. $b \rightarrowtail p \rightarrow \neg f$	*LLE(8),* $\models b \Leftrightarrow (p \wedge b) \vee (\neg p \wedge b)$
10. $b \wedge f \wedge (p \rightarrow \neg f) \rightarrowtail b \wedge f \wedge (p \rightarrow \neg f)$	*REF*
11. $b \wedge f \wedge (p \rightarrow \neg f) \rightarrowtail f \wedge (p \rightarrow \neg f)$	*RW(10),* $\models b \wedge f \wedge (p \rightarrow \neg f) \rightarrow f \wedge (p \rightarrow \neg f)$
12. $b \wedge (p \rightarrow \neg f) \rightarrowtail f \wedge (p \rightarrow \neg f)$	*CUT(1, 11)*
13. $b \rightarrowtail f \wedge (p \rightarrow \neg f)$	*CUT(12, 9)*
14. $b \rightarrowtail \neg p$	*RW(13),* $\models f \wedge (p \rightarrow \neg f) \rightarrow \neg p$

Адамс (енг. *Ernest Wilcox Adams*) [1], касније Перл (енг. *Judea Pearl*) [92] и Лехман и Магидор (енг. *Daniel Lehmann* и енг. *Menachem Magidor*) [63] су предложили вероватносну интерпретацију дифолта: дифолти $\alpha \rightarrowtail \beta$ означавају да 'вероватноћа од β за дато α је *врло близу* 1'. Прецизно значење израза *врло близу* се односи на вероватноће чије су вредности из не-Архимедовог поља, тј. уређеног поља које садржи инфинитезимале: елемент x је инфинитезимала ако $|x| < 1/n$, за сваки позитиван цео број n. Два елемента x и y су *бесконачно блиска*, у ознаци $x \approx y$, ако је $x - y$ инфинитезимала. То значи да је x инфинитезимала ако и само ако је $x \approx 0$.

Користећи погодно елементарно проширење ${}^*\mathbb{R}$ стандардних реалних бројева, тј. ослањајући се на Робинсонове (енг. *Abraham Robinson*) основне резултате нестандардне анализе, ${}^*\mathbb{R}$-вероватносни модел се може дефинисати као тројка $(W, \mathcal{F}, \mu)$, где је W скуп могућих светова (истинитосна интерпретација исказних слова), $\mathcal{F}$ је поље подскупова од W које садржи све скупове одређене исказним формулама, и $\mu : \mathcal{F} \to {}^*\mathbb{R}$ је коначна адитивна ${}^*\mathbb{R}$-вредносна вероватносна мера. Дифолт $\alpha \rightarrowtail \beta$ важи у ${}^*\mathbb{R}$-вероватносном моделу ако је:

- или вероватноћа формуле α нула,
- или условна вероватноћа формуле β за дато α бесконачно близу 1.

Уобичајено је да се каже да су дифолти дати у ε-семантици. База дифолта Δ ε-подразумева дифолт $\alpha \rightarrowtail \beta$, у ознаци $\Delta \vdash_\varepsilon \alpha \rightarrowtail \beta$, ако је могуће обезбедити да је $P(\beta \mid \alpha)$ скоро 1, узимајући да су вероватноће дифолта из базе Δ скоро 1.

Испоставља се да је ${}^*\mathbb{R}$, из много разлога, компликован простор за коришћење у овом истраживању и често се може заменити једноставнијим не-Архимедовим пољем. Потрага за минималним не-Архимедовим вредностима вероватноћа навела је многе ауторе [38, 39, 91] да разматрају уређено поље $\mathbb{R}(\varepsilon)$. Ово поље је најмање поље које се добија када се реалним бројевима дода јединствена инфинитезимала ε. $\mathbb{Q}(\varepsilon)$ је још један пример не-Архимедовог поља који представља проширење рационалних бројева.

3.2 Инфинитезимални рачун

Сваки елемент поља $\mathbb{Q}(\varepsilon)$ је рационални израз $\frac{p(\varepsilon)}{q(\varepsilon)}$, где су $p(\varepsilon)$ и $q(\varepsilon)$ полиноми по ε над $\mathbb{Q}$, и $q(\varepsilon)$ није идентички једнако са нулом. Два рационална израза $\frac{p(\varepsilon)}{q(\varepsilon)}$ и $\frac{p_1(\varepsilon)}{q_1(\varepsilon)}$ су једнака ако полиноми $p(\varepsilon)q_1(\varepsilon)$ и $p_1(\varepsilon)q(\varepsilon)$ имају исте не-нула коефицијенте. Ови рационални изрази се сабирају и множе на уобичајени начин. Оваквим начином дефинисања $\mathbb{Q}(\varepsilon)$ има особине поља. Сваки елемент η поља $\mathbb{Q}(\varepsilon)$ се може представити у нормализованом облику:

$$(*) \qquad \eta = \frac{a\varepsilon^k + \sum_{i=k+1}^{n} a_i \varepsilon}{1 + \sum_{j=1}^{m} b_j \varepsilon^j}, k < n, 0 < m$$

за неки јединствени цео број k и јединствено водећи коефицијент a такав да је $a \neq 0$ осим у случају када је $\eta = 0$. На основу дефинисаног уређења $<$ на $\mathbb{Q}(\varepsilon)$ важи да је $\eta > 0$ ако и само ако $a > 0$. Ово значи да, пошто је ε инфинитезимала, $\mathbb{Q}(\varepsilon)$ је уређено не-Архимедово поље. При томе важи да је $\mathbb{Q} \subsetneqq \mathbb{Q}(\varepsilon)$.

У наставку, пажња ће бити усмерана на $\mathbb{Q}(\varepsilon)$-вероватносни простор $(W, \mathcal{F}, \mu)$ са коначном адитивном вероватносном мером $\mu : \mathcal{F} \to \mathbb{I}$, где је $\mathbb{I}$ јединични интервал од $\mathbb{Q}(\varepsilon)$.

3.3 Моделирање дифолт закључивања помоћу вероватносних логика

У делу 2.3 је описана логика $LPCP$ са приближним условним оператором вероватноће. У овој логици, дифолт $\alpha \rightarrowtail \beta$ се може записати као формула облика $CP_{\approx 1}(\beta, \alpha)$. Наравно, коначна база дифолта $\Delta = \{\alpha_1 \rightarrowtail \beta_1, \ldots, \alpha_m \rightarrowtail \beta_m\}$ је представљена скупом $\overline{\Delta} = \{CP_{\approx 1}(\beta_1, \alpha_1), \ldots, CP_{\approx 1}(\beta_m, \alpha_m)\}$. Показано је да овакав приступ даје карактеризацију система $\mathbf{P}$.

Теорема 3.3.1. *За сваку базу дифолта Δ и сваки дифолт $\alpha \rightarrowtail \beta$:*

$$\Delta \mid\!\sim_{\mathbf{P}} \alpha \rightarrowtail \beta \text{ акко } \Delta \vdash_{\mathrm{Ax}_{LPCP}} \alpha \rightarrowtail \beta.$$

Коначна база дифолта $\Delta = \{\alpha_1 \rightarrowtail \beta_1, \ldots, \alpha_m \rightarrowtail \beta_m\}$ се може посматрати као формула $\bigwedge \overline{\Delta} = \bigwedge_{i=1}^{m} CP_{\approx 1}(\beta_i, \alpha_i)$. Користећи теорему дедукције и потпуности за $LPCP$ ([101]), добија се:

$\Delta \mid\!\sim_{\mathbf{P}} \alpha \rightarrowtail \beta$
акко $\vdash_{\mathrm{Ax}_{LPCP}} \bigwedge \overline{\Delta} \Rightarrow CP_{\approx 1}(\beta, \alpha)$
акко $\bigwedge \overline{\Delta} \Rightarrow CP_{\approx 1}(\beta, \alpha)$ је ваљано у односу на класу $\mathbb{Q}(\varepsilon)$-вероватносних простора
акко $\neg \left(\bigwedge \overline{\Delta} \Rightarrow CP_{\approx 1}(\beta, \alpha) \right)$ није задовољиво ни у једном $\mathbb{Q}(\varepsilon)$-вероватносном простору
акко $\bigwedge \overline{\Delta} \wedge \neg\, CP_{\approx 1}(\beta, \alpha)$ није задовољиво ни у једном $\mathbb{Q}(\varepsilon)$-вероватносном простору.

У поглављу 2.3 је описана процедура свођења проблема задовољивости на проблем линеарног програмирања и тиме доказана одлучивост тог проблема. Користећи исту процедуру, у наставку ће бити илустровано како се проблем задовољивости претходно добијене формуле облика

$$\overline{\delta} = \bigwedge_{i=1}^{m} CP_{\approx 1}(\beta_i, \alpha_i) \wedge \neg\, CP_{\approx 1}(\beta, \alpha).$$

своди на проблем проналажења решења система линеарних (не)једначина.

Нека су $p_1,\ldots,p_n$ сва исказна слова која се јављају у исказним формулама које сачињавају дифолте. Атом a је формула $\pm p_1 \wedge \ldots \wedge \pm p_n$. Са At ће бити означен скуп свих атома. Очигледно је да је $|At| = 2^n$.

Скуп исказних слова је коначан, па ће и у овом случају бити разматран $\mathbb{Q}(\varepsilon)$-вероватносни модел $(At, \mathcal{P}(At), \mu)$, где је $\mu : \mathcal{P}(At) \to \mathbb{I}$ коначна адитивна вероватносна мера, а x_i означава меру атома $a_i \in At$, $i = 1, \ldots, 2^n$. Дакле, формула $\overline{\delta}$ је задовољива ако и само ако систем

$$(*) \qquad \begin{cases} \sum_{i=1}^{N} x_i = 1, \\ x_1 \geq 0, \ldots, x_N \geq 0, \\ \sum_{a_i \in At, a_i \models \alpha_1} x_i > 0, \ldots, \sum_{a_i \in At, a_i \models \alpha_m} x_i > 0, \sum_{a_i \in At, a_i \models \alpha} x_i > 0, \\ \frac{\sum_{a_i \in At, a_i \models \alpha_1 \wedge \beta_1} x_i}{\sum_{a_i \in At, a_i \models \alpha_1} x_i} \approx 1, \ldots, \frac{\sum_{a_i \in At, a_i \models \alpha_m \wedge \beta_m} x_i}{\sum_{a_i \in At, a_i \models \alpha_m} x_i} \approx 1, \\ \frac{\sum_{a_i \in At, a_i \models \alpha \wedge \beta} x_i}{\sum_{a_i \in At, a_i \models \alpha} x_i} \not\approx 1, \end{cases}$$

има решење у $\mathbb{Q}(\varepsilon)$.

Аналогно разматрању у поглављу 2.3, систем $(*)$ има решење у $\mathbb{Q}(\varepsilon)$ ако и само ако систем

$$(\star) \qquad \begin{cases} \sum_{i=1}^{N} x_i = 1, \\ x_1 \geq 0, \ldots, x_N \geq 0, \\ \sum_{a_i \in At, a_i \models \alpha_1} x_i > 0, \ldots, \sum_{a_i \in At, a_i \models \alpha_m} x_i > 0, \sum_{a_i \in At, a_i \models \alpha} x_i > 0, \\ 0 \leq 1 - \frac{\sum_{a_i \in At, a_i \models \alpha_1 \wedge \beta_1} x_i}{\sum_{a_i \in At, a_i \models \alpha_1} x_i} < K\varepsilon, \ldots, 0 \leq 1 - \frac{\sum_{a_i \in At, a_i \models \alpha_m \wedge \beta_m} x_i}{\sum_{a_i \in At, a_i \models \alpha_m} x_i} < K\varepsilon, \\ 1 - \frac{\sum_{a_i \in At, a_i \models \alpha \wedge \beta} x_i}{\sum_{a_i \in At, a_i \models \alpha} x_i} > \frac{1}{K}, \end{cases}$$

има решење у ${}^*\mathbb{R}$, где су ε и K погодно изабрни елементи поља ${}^*\mathbb{R}$.

Очигледно је да је систем $(\star)$ еквивалентан систему линеарних неједнакости облика

$$s_1 x_{i_1} + \ldots s_\ell x_{i_\ell} \, \rho \, 0,$$

где је $\ell \leq N$, $1 \leq i_1, \ldots, i_\ell \leq N$, s_j рационални израз по ε и K, и $\rho \in \{\leq, <, >, \geq\}$. На добијен систем се може применити Фурије-Моцкин процедура елиминације[1], која итеративно трансформише систем у еквивалентан систем у коме је елиминисана непозната x_i. Током примене процедуре, коефицијенти у неједнакостима остају рационални изрази по ε и K. Када су све непознате елиминисане, потребно је проверити да ли су релације међу изразима по ε и K тачне.

Пример 3.3.1. *У примеру 3.1.1, је показано да је* $\{b \rightarrowtail f, p \rightarrowtail b, p \rightarrowtail \neg f\} \mathrel{|\!\sim_{\mathbf{P}}} b \rightarrowtail \neg p$. *Сада ће исто бити показано на други начин, користећи претходно описану процедуру.*

[1] Детаљан опис процедуре дат је у поглављу А.1

Постоји осам атома које треба разматрати:

$$a_1 = b \wedge p \wedge f \quad a_2 = b \wedge p \wedge \neg f \quad a_3 = b \wedge \neg p \wedge f \quad a_4 = b \wedge \neg p \wedge \neg f$$
$$a_5 = \neg b \wedge p \wedge f \quad a_6 = \neg b \wedge p \wedge \neg f \quad a_7 = \neg b \wedge \neg p \wedge f \quad a_8 = \neg b \wedge \neg p \wedge \neg f$$

Потребно је показати да формула

$$CP_{\approx 1}(f, b) \wedge CP_{\approx 1}(b, p) \wedge CP_{\approx 1}(\neg f, p) \wedge CP_{\not\approx 1}(\neg p, b)$$

није задовољива ни у једном од $\mathbb{Q}(\varepsilon)$-вероватносних простора, тј. да систем:

$$(*) \qquad \begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 = 1, \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0, x_6 \geq 0, x_7 \geq 0, x_8 \geq 0, \\ x_1 + x_2 + x_3 + x_4 > 0, x_1 + x_2 + x_5 + x_6 > 0, \\ \frac{x_1+x_3}{x_1+x_2+x_3+x_4} \approx 1, \frac{x_1+x_2}{x_1+x_2+x_5+x_6} \approx 1, \frac{x_2+x_6}{x_1+x_2+x_5+x_6} \approx 1, \\ \frac{x_3+x_4}{x_1+x_2+x_3+x_4} \not\approx 1, \end{cases}$$

нема решење у $\mathbb{Q}(\varepsilon)$. Први корак описане процедуре је елиминација релације $\approx$, тј. транформација претходног система у систем:

$$(\star) \qquad \begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 = 1, \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0, x_6 \geq 0, x_7 \geq 0, x_8 \geq 0, \\ x_1 + x_2 + x_3 + x_4 > 0, x_1 + x_2 + x_5 + x_6 > 0, \\ 1 - \frac{x_1+x_3}{x_1+x_2+x_3+x_4} \geq 0, 1 - \frac{x_1+x_2}{x_1+x_2+x_5+x_6} \geq 0, 1 - \frac{x_2+x_6}{x_1+x_2+x_5+x_6} \geq 0, \\ 1 - \frac{x_1+x_3}{x_1+x_2+x_3+x_4} < K\varepsilon, 1 - \frac{x_1+x_2}{x_1+x_2+x_5+x_6} < K\varepsilon, 1 - \frac{x_x+x_6}{x_1+x_2+x_5+x_6} < K\varepsilon, \\ 1 - \frac{x_3+x_4}{x_1+x_2+x_3+x_4} > \frac{1}{K}, \end{cases}$$

који је еквивалентан са системом линеарних неједнокости:

$$(\star) \qquad \begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 = 1, \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0, x_6 \geq 0, x_7 \geq 0, x_8 \geq 0, \\ x_1 + x_2 + x_3 + x_4 > 0, x_1 + x_2 + x_5 + x_6 > 0, \\ x_2 + x_4 \geq 0, x_5 + x_6 \geq 0, x_1 + x_5 \geq 0, \\ x_2 + x_4 < K\varepsilon(x_1 + x_2 + x_3 + x_4), \\ x_5 + x_6 < K\varepsilon(x_1 + x_2 + x_5 + x_6), \\ x_1 + x_5 < K\varepsilon(x_1 + x_2 + x_5 + x_6), \\ x_1 + x_2 > \frac{1}{K}(x_1 + x_2 + x_3 + x_4). \end{cases}$$

Фурије-Моцкин процедура елиминације се може применити на стандарадан начин и процедура ће се завршити са закључком да систем нема решење.

Решавање последњег система је веома дугачко и споро, тако да ће бити разматран једноставнији систем у коме су изостављени неки „небитни" атоми a_1, a_5, a_6, a_7, a_8 ($x_1 = x_5 = x_6 = x_7 = x_8 = 0$) и избегнуте редудантне неједнакости. Уствари, из разматрања су искључена створења која нису птице и пингвини који

лете.

$$\begin{cases} x_2+x_3+x_4=1 \\ x_2>0, x_3\geq 0, x_4\geq 0 \\ x_2+x_4<K\varepsilon(x_2+x_3+x_4) \\ x_2>\frac{1}{K}(x_2+x_3+x_4) \end{cases} \Leftrightarrow \begin{cases} 1-x_3-x_4 \leq x_2 \\ x_2 \leq 1-x_3-x_4 \\ 0<x_2 \\ x_2<\frac{K\varepsilon}{1-K\varepsilon}x_3-x_4 \\ \frac{1}{K-1}x_3+\frac{1}{K-1}x_4<x_2 \\ 0\leq x_3 \\ 0\leq x_4 \end{cases}$$

$$\begin{cases} 1-x_3-x_4\leq 1-x_3-x_4 \\ 1-x_3-x_4<\frac{K\varepsilon}{1-K\varepsilon}x_3-x_4 \\ 0<1-x_3-x_4 \\ 0<\frac{K\varepsilon}{1-K\varepsilon}x_3-x_4 \\ \frac{1}{K-1}x_3+\frac{1}{K-1}x_4<1-x_3-x_4 \\ \frac{1}{K-1}x_3+\frac{1}{K-1}x_4<\frac{K\varepsilon}{1-K\varepsilon}x_3-x_4 \\ 0\leq x_3 \\ 0\leq x_4 \end{cases} \Leftrightarrow \begin{cases} 1-K\varepsilon < x_3 \\ x_3 < 1-x_4 \\ \frac{1-K\varepsilon}{K\varepsilon}x_4 < x_3 \\ x_3 < \frac{K-1}{K}-x_4 \\ x_3 < \frac{-K+K^2\varepsilon}{1-K^2\varepsilon}x_4 \\ 0\leq x_3 \\ 0\leq x_4 \end{cases}$$

$$\begin{cases} 1-K\varepsilon<1-x_4 \\ 1-K\varepsilon<\frac{K-1}{K}-x_4 \\ 1-K\varepsilon<\frac{-K+K^2\varepsilon}{1-K^2\varepsilon}x_4 \\ \frac{1-K\varepsilon}{K\varepsilon}x_4<1-x_4 \\ \frac{1-K\varepsilon}{K\varepsilon}x_4<\frac{K-1}{K}-x_4 \\ \frac{1-K\varepsilon}{K\varepsilon}x_4<\frac{-K+K^2\varepsilon}{1-K^2\varepsilon}x_4 \\ 0<1-x_4 \\ 0<\frac{K-1}{K}-x_4 \\ 0<\frac{-K+K^2\varepsilon}{1-K^2\varepsilon}x_4 \\ 0\leq x_4 \end{cases} \Leftrightarrow \begin{cases} x_4 < K\varepsilon \\ x_4 < \frac{-1+K^2\varepsilon}{K} \\ x_4 < \frac{(1-K\varepsilon)(1-K^2\varepsilon)}{-K+K^2\varepsilon} \\ x_4 < (K-1)\varepsilon \\ x_4 < 0 \\ x_4 < 1 \\ x_4 < \frac{K-1}{K} \\ 0 \leq x_4 \end{cases}$$

Очигледно је да неједнакости $0<\frac{-1+K^2\varepsilon}{K}$ *и* $0<0$ *нису тачне.*

И у овом случају се, на основу процедуре трансформације, проблем задовољивости своди на проблем линеарног програмирања који се може дефинисати на следећи начин:

$$\begin{aligned} & \min z(x) \\ s.t.\quad & \textstyle\sum_{i=1}^{2^n} c_{ij}x_i \ \rho\ 0 \quad j=1,2,\dots L, \quad \rho\in\{\geqslant,>,<,\leqslant\} \\ & \textstyle\sum_{i=1}^{2^n} x_i = 1 \\ & x_i \geqslant 0 \qquad\qquad i=1,2,\dots,2^n \end{aligned}$$

где се $\sum_{i=1}^{2^n} c_{ij}x_i\ \rho\ 0$ добија поједностављењем неједнакости $(\star)$, L је број тих неједнакости и све вредности $c_{ij}, x_i (i=1,2,\dots,2^n, j=1,2,\dots,L)$ припадају Хардијевом

пољу, тј. то су рационални изрази по ε и K. Функција циља $z(x)$ се дефинише као

$$z(x) = \sum_{i=1}^{L} d_i(x),$$

где је

$$d_i(x) = \begin{cases} (\sum_{i=1}^{2^n} c_{ij} x_i)^2 & \text{ако } j\text{-та неједнакост није задовољена,} \\ 0 & \text{у супротном,} \end{cases}$$

У поглављу 6 ће функција циља бити детаљније разматрана.

Глава 4

Метахеуристичке методе

У овом поглављу пажња ће бити посвећена хеуристичком приступу решавања проблема. Најпре ће бити дефинисан проблем оптимизације, а затим ће бити представљене неке хеуристичке методе које су коришћене за решавање проблема задовољивости формула у вероватносним логикама. Посебна пажња ће бити посвећена методи оптимизације колонијом пчела која је коришћена за решавање проблема задовољивости фомула логике $LPCP$ и дифолтне логике.

4.1 Проблем оптимизације

У општем случају, задатак оптимизације се дефинише на следећи начин [13, 15, 90, 115]:

Дефиниција 4.1.1. *Нека је $f : S \to \mathbb{R}$ реална функција дефинисана на скупу S и нека је $X \subseteq S$ неки задати скуп. Проблем је наћи*

$$\min f(x)$$

под условом (ограничењем)

$$x \in X.$$

Проблем оптимизације се назива *проблем комбинаторне* или *дискртне оптимизације* уколико је скуп X коначан или пребројив. Ако скуп X не садржи дискретне вредности тада се проблем оптимизације назива *континуална оптимизација*, док се у супротном проблем оптимизације назива *мешовита оптимизација*. Скуп S садржи потенцијална решења проблема, па се назива *простор претраге* или *простор решења*. Скуп X садржи допустива решења, па се сходно томе, назива *простор допустивих решења*.

У литертаури се често може наћи и детаљнија дефиниција оптимизационог проблема у облику [122]:

Дефиниција 4.1.2.

$$\begin{array}{lll} \min\limits_{\mathbf{x}\in S, S\subseteq\mathbb{R}^d} & f_i(\mathbf{x}), & (i=1,2,\ldots,M), \\ \textit{под условом} & h_j(\mathbf{x})=0, & (j=1,2,\ldots,J), \\ & g_k(\mathbf{x})\leqslant 0, & (k=1,2,\ldots,K), \end{array} \tag{4.1.1}$$

где су $f_i(\mathbf{x})$, $h_j(\mathbf{x})$ *и* $g_k(\mathbf{x})$ *функције над вектором* $\mathbf{x}=(x_1,x_2,\ldots,x_d)^T$.

Компоненте x_i вектора $\mathbf{x}$ могу бити реалне, дискретне или комбинација ова два типа. Функције $f_i(\mathbf{x})$ где је $i=1,2,\ldots,M$ се називају *функције циља* (енг. *objective functions*). Једнакости за $h_j(\mathbf{x})$ и неједнакости за $g_k(\mathbf{x})$ називају се *ограничења* (енг. *constraints*) и дефинишу простор допустивих решења X. Важно је напоменути да релација у неједнакостима може бити и $\geqslant 0$, као и да се циљеви могу формулисати као проблеми максимизације. У екстремним случајевима, функција циља не мора да буде дефинисана, већ су дефинисана само ограничења. Овакви проблеми се називају *проблеми допустивости (изводљивости, оправданости)* (енг. *feasibility problem*), пошто је код ових проблема било које могуће (изводљиво) решење уједно и оптимално.

Оптимизациони проблеми се могу класификовати према броју циљева у две категорије: оптимизациони проблеми са једним циљем (енг. *single objective*) за $M=1$ и оптимизациони проблеми више циљева (енг. *multiobjective*) за $M>1$. Реални проблеми оптимизације су, углавном, са више циљева, међутим оптимизациони алгоритми за оптимизацију са једним циљем се могу прилагодити и за решавање вишециљних оптимизационих проблема, па ће, због једноставности, у наставку бити разматрана само оптимизација са једним циљем (у складу са тим функција циља биће означавана са f).

Дефиниција 4.1.3. *Решење* $\mathbf{x}^*\in S$ *је* ***глобални оптимум*** *ако одговара мањој вредности функције циља од свих осталих решења из простора решења, тј. ако* $\forall\mathbf{x}\in S, f(\mathbf{x}^*)\leqslant f(\mathbf{x})$.

У складу са овом дефиницијом, главни циљ у решавању оптимизационог проблема је проналажење глобалног оптимума $\mathbf{x}^*$. За дати проблем може постојати више глобалних оптималних решења, па се за неке проблеме, као задатак, може дефинисати и проналажење свих глобалних оптималних решења.

Класификација оптимизационих проблема може се вршити на још неколико начина. Сем већ поменутих које зависе од скупа X и броја циљева, класификација се може вршти и на основу броја ограничења $J+K$. Уколико ограничења не постоје, $J=K=0$, тада се разматра *оптимизација без ограничења* (енг. *unconstrained optimization*). Уколико је $K=0$ и $J\geqslant 0$, тада се проблем оптимизације назива *једнакосно ограничени проблем* (енг. *equality-constrained*), док се у случају да је $J=0$ и $K\geqslant 0$ проблем назива *неједнакосно ограничени проблем* (енг. *inequality-constrained*).

У литератури се често срећу дефиниције само неједнакосно ограничених оптимизационих проблема, пошто се једнакост $h(\mathbf{x}) = 0$ може записати помоћу две неједнакости $h(\mathbf{x}) \leqslant 0$ и $h(\mathbf{x}) \geqslant 0$.

Према типу функција циља и ограничења, оптимизациони проблем може бити *линеаран* или *нелинеаран*. Линеарни оптимизациони проблеми се могу записати у облику:

$$\begin{aligned} \min f(x) &= c^T \cdot x \\ A \cdot x &\leqslant b \end{aligned} \tag{4.1.2}$$

што указује на то да су функција циља и ограничења линеарне функције, па се могу представити матрицом A и векторима b и c. Уколико су елементи матрице A и вектора b и c реални бројеви, посматрани оптимизациони проблем се назива проблем *линеарног програмирања* (енг. *Linear Programming*, LP). Специјално, ако су елементи матрице A и вектора b и c цели бројеви, разматра се проблем *целобројног линеарног програмирања* (енг. *Integer Linear Programming*, ILP), а у случају да су вредности које се појављују и реални и цели бројеви, тада се говори о проблему *мешовитог целобројног линеарног програмирања* (енг. *Mixed Integer Linear Programming*, MILP). Уколико функција циља и/или ограничења нису линарне, проблем оптимизације се назива *нелинеарна оптимизација*.

Још један начин поделе оптимизационих проблема је на *детерминистичке* и *стохастичке* оптимизационе проблеме. Уколико су параметри, којима се оптимизациони проблем описује, тачно познати, проблем је детерминистичке природе, док у случају да су познате само приближне вредности параметара или њихова процена, проблем је стохастичке природе. Обзиром на неизвесност која је присутна у стохастичким проблемима, вредност функције циља није могуће тачно одредити, нити прецизно проверити да ли су нека од ограничења нарушена или не, па се ови проблеми моделирају уз помоћ додатних техника, као што су Марковљеви процеси, вероватносне логике и фази скупови.

4.2 Класификација метода

Проналажење оптималног решења, за многе реалне оптимизационе проблеме, је често немогуће. У пракси је, углавном, довољно да се пронађе „добро" решење коришћењем хеуристичких или метахеуристичких алгоритама. Хеуристички алгоритми обезбеђују „прихватљиво" решење комплексног проблема за релативно кратко време. За разлику од егзактних метода, хеуристичке методе не гарантују оптималност добијеног решења, па чак ни оцену удаљености генерисаног решења од оптималног.

Хеуристичеке методе су развијане за решавање конкретних проблема, док је појам *метахеуристике* увео Гловер у свом раду [29]. Метахеуристичке методе се могу дефинисати као методологије (шаблони) вишег нивоа који се могу користити као опште стратегије за дефинисање хеуристике за решавање конкретног оптимизационог проблема [115].

Хеуристичке методе се могу поделити у неколико група [15]:

(i) Конструктивне хеуристике. Ова група метода се заснива на постепеној конструкцији (изградњи) решења, при чему се користе специфична знања сваког појединачног задатка.

(ii) Хеуристике итеративног побољшавања. Ова група се назива и *методе локалног претраживања*. При решавању проблема, полази се од произвољног почетног решења, које се постепено побољшава претраживањем њему „блиских" решења. Процес се понавља све док у „близини" текућег решења постоји боље решење од текућег.

(iii) Хеуристике математичког програмирања. Посматрани проблем се формулише као задатак математичког програмирања, а затим се решава приближним методама.

(iv) Декомпозиционе хеуристике. Почетни проблем се дели на више мањих подпроблема, који се засебно решавају.

(v) Хеуристике базиране на подели допустивог скупа. Допустиви скуп решења се подели на више подскупова, па се претраживањем решења у сваком од њих тражи најбоље решење. За хеуристичко решење проблема се узима оно решење за које функција циља има најмању вредност.

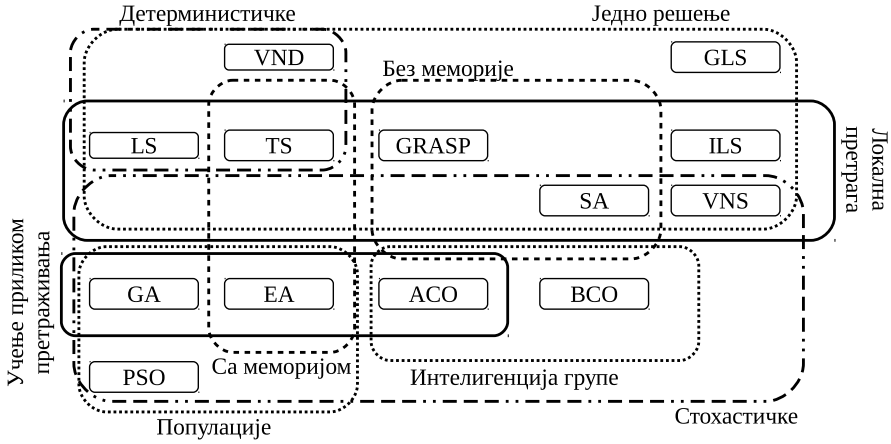
(vi) Хеуристике базиране на рестрикцији допустивог скупа. Рестрикцијом допустивог скупа решења се елиминишу одређени подскупови и тако се смањује простор претраге.

(vii) Хеуристике базиране на релаксацији. Релаксацијом се допустиви скуп решења проширује, али тако да се омогући једноставније решавање новог проблема.

У решавању неког проблема, наравно, могу се комбиновати различити типови хеуристика.

Обзиром да је у последње две деценије развијен велики број метахеуристичких метода, постоје и многи други начини њихове класификације. Као пример различитих класификација дата је слика 4.1 која садржи само репрезентативне примере за сваку класу метахеуристичких метода и није потпуно прецизна пошто су класификације бројне и међусобно се преплићу.

Један начин класификација метахеуристика је на основу стратегије претраге. Један тип стратегија претраге је поправљање решења алгоритмом локалне претраге. Метахеуристике овог типа су симулирано каљење (енг. *Simulated Annealing*, SA) [56], табу претраживање (енг. *Tabu Search*, TS) [29], итеративно локално претраживање (енг. *Iterated Local Search*, ILS) [67], метода променљивих околина (енг. *Variable Neighborhood Search*, VNS) [80], похлепна стохастичка прилагодљива процедура претраживања (енг. *Greedy Randomized Adaptive Search Procedure*, GRASP), [24]. Други тип стратегије претраживања садржи компоненту учења приликом претраживања,



Слика 4.1: Класификација метахеуристика

каква се среће код метахеуристика базираних на колонији мрава (енг. *Ant Colony Optimization*, ACO) [21], генетском алгоритму (енг. *Genetic Algorithm*, ГА) [48] и другим еволутивним алгоритмима (енг. *Evolutionary Algorithm*, EA).

Следећа врста класификације је претраживање на бази једног решења наспрам претрага базираних на популацији. Приступ који се заснива на коришћењу једног решења базира се на модификовању и поправљању једног кандидата за решење, такве метахеуристике су SA, ILS, VNS, вођена локална претрага (енг. *Guided Local Search*, GLS) [3]. Приступ базиран на популацији разматра и поправља више кандидата за решење, често користећи карактеристике популације у претрази. Такве метахеуристике су EA, GA, оптимизацијаројевима честица (енг. *Particle Swarm Optimization*, PSO) [96]. Још једна катергорија је базирана на интелигенцији групе (енг. *Swarm intelligence*) које користе колективно понашања самоорганизованих агената у популацији или року. Такве метахеуристике су ACO, PSO, колоније пчела (енг. *Bee Colony Optimization*, BCO и енг. *Bee Swarm Optimization*, BSO) [51, 116].

Метахеуристике се могу класификовати и на основу коришћења меморије. Неки метахеуристички алгоритми не користе информације добијене током претраге, попут GRASP и SA метода. Док друге метахеуристике користе меморију која садржи информације добијене током претраге, попут TS и различитих EA.

Такође, могу се разматрати детерминистичке наспрам стохастичких метода. Код детерминистичких метода проблем се решава доношењем детерминистистичких одлука. Такве метахеуристике су, на пример, LS, TS и метода променљивог спуста (енг. *Variable Neighborhood Descent*, VND). Код стохастичких метахеуристика користе се случајна правила током претраге, каква се јављају код, на пример, SA, VNS, BCO и EA. Код детерминистистичких метода коришћење истог иницијалног решења ће увек довести до истог коначног решења, док се код стохастичких метода може доћи до различитих коначних решења коришћењем истог иницијалног решења.

Као додатак поменутим алгоритмима, постоје хибридне и паралелне метахеу-

ристичке методе. Хибридне метахеуристике су оне код којих се нека метахеуристика комбинује са другом оптимизационом методом (попут линеарног програмирања, машинског учења и слично) или другом метахеуристичком методом. Обе компоненте хибридне метахеуристике раде конкуретно и размењују информације током претраге. Паралелне хеуристике користе технике паралелног програмирања за стартовање више метахеуристичких претрага паралелно. Овако стартовани паралелни процеси могу да сарађују и размењују информације у потрази за решењем.

Имплементација великог броја метахеуристичких метода заснована је на појму допустивог решења и дефиниције његове околине (у циљу примене итеративног побољшавања). Да би се дефинисала околина неког решења, мора се увести појам растојања између две тачке [2].

Дефиниција 4.2.1. *Нека је X произвољан скуп тачака. Ако се сваком уређеном пару (x, y) тачака из X придружи реалан број $d(x, y)$ који има особине:*

1. $0 \leqslant d(x, y) < +\infty$,
2. $d(x, y) = 0 \Leftrightarrow x = y$,
3. $d(x, y) = d(y, x)$,
4. $d(x, y) \leqslant d(x, z) + d(z, y)$,

*каже се да је скуп X снабдевен **метриком** d. Такав скуп X, тј. уређен пар (X, d), назива се **метричким простором**. Вредност $d(x, y)$ представља **растојање** између тачака x и y.*

Дефиниција 4.2.2. *Скуп $\mathcal{N}_d(x) \subseteq X$ свих тачака на растојању d од неке изабране тачке $x \in X$ назива се d-**околина тачке** x. Елементи скупа $\mathcal{N}_d(x)$ називају се d-**суседи** тачке x.*

Општија дефиниција може обухаватати све тачке на растојању мањем или једнаком d. Уколико је вредност d унапред позната и фиксирана, онда се може говорити о околини тачке x, не наглашавајући да се ради о d-околини.

У оптимизационим проблемима често се користи мање формална дефиниција околине и суседа неког решења x [95]. Под околином решења x се подразумева $\mathcal{N}(x) \subseteq X$ који је придружен решењу x по неком правилу. То је скуп решења добијених од x применом неке трансформације. Дефиниција трансформације зависи од методе која се користи, као и од њене конкретне имплементације. Обично се подразумева да решење x не припада својој околини, тј. $x \notin \mathcal{N}(x)$. У овом случају се, такође, могу посматрати d-околине неког решења, при чему је d природан број и $\mathcal{N}_d(x)$ представља скуп свих решења која се добијају применом d пута (исте) трансформације на решење x.

Дефиниција 4.2.3. *Решење $x \in S$ назива се* ***локални минимум*** *неког оптимизационог проблема уколико не постоји $x' \in \mathcal{N}(x) \subseteq S$ такво да важи $f(x') < f(x)$.*

Локални минимуми се називају субоптимална решења. Обзиром на сложеност оптимизационих проблема, све метахеуристичке методе су усмерене ка тражењу што бољих субоптималних решења у што краћем времену.

4.3 Преглед метахеуристичких метода

4.3.1 Локално претраживање

Метода итеративног побољшавања код које је претраживање ограничено на унапред дефинисану околину решења у потрази за локалним минимумом назива се локално претраживање (енг. *Local Search*, LS) [15]. Прецизније, за свако $x' \in \mathcal{N}(x)$ у околини неког почетног решења x, израчунава се вредност функције циља $f(x')$ и уколико је $f(x') < f_{min}$, чува се ново $x_{min} = x'$ и $f_{min} = f(x')$, уколико јеx_{min} различито од x. Када се „обиђу" сва решења у околини $\mathcal{N}(x)$, наставља се претрага у околини $\mathcal{N}(x_{min})$. Испитивање свих суседа неког решења x_{min} назива се претраживање околине (енг. *neighborhood exploration*, NE). NE представља једну итерацију локалног претраживања. Процес локалног претраживања одвија се у итерацијама и зауставља се када у околини $\mathcal{N}(x_{min})$ не постоји боље решење.

Понекад је сувише споро, па чак и немогуће претражити целу околину $\mathcal{N}(x_{min})$. Стога постоје разни начини (хеуристике) како да се одреде суседи који „обећавају" поправљање вредности функције циља, тј. да се на неки начин редукује величина околине која се претражује.

Основни недостатак локалног претраживања је што се зауставља при проналаску првог локалног минимума који не мора бити (и најчешће није) глобални минимум, а то увелико зависи од почетног решења.

Примена локалног претраживања у испитивању задовољивости формула вероватносне логике

Проблем задовољивости у вероватносним логикама се често ознчава са PSAT. У раду [52] је описана процедура заснована на LS алгоритму која је примењена на следећи проблем: Дат је скуп $S = \{C_1, \ldots, C_m\}$ од m исказних клауза (исказних дисјункција), у којима се појављује n исказних слова $x_1, \ldots, x_n$. Такође, дато је и m вероватноћа $0 \leqslant p_1, \ldots, p_m \leqslant 1$, по једна за сваку клаузу. Посматра се простор догађаја који садржи свих 2^n истинитосних валуација. Вероватносна расподела је скуп од 2^n вероватноћа $\pi_j \geqslant 0$, $j = 1, \ldots, 2^n$ чија је сума 1. Вероватносна расподела π задовољава дати скуп клауза и вероватноћа ако важи да је за свако $i = 1, \ldots, m$ сума π_j за све истинитосне валуације a_j које задовољавају C_i једнака p_i. Ово се може

алгебарски записати дефинисањем $m \times 2^n$ матрице A такве да је елемент (i, j) једнак 1 ако a_j задовољава C_i, у супротном 0. Описани захтеви се сада могу записати као:

$$A\pi = p. \qquad \pi \geqslant 0.$$

Задатак је испитати да ли овакво π постоји.

У раду [52] се полази од чињенице да су описани PSAT и MAXSAT дуални проблеми ([28]), при чему је опис општег MAXSAT проблема исказне логике дат са:

- Дат је скуп од m клауза са n исказних слова. За сваку клаузу C_i дата је целобројна тежина w_i и дата је вредност W.
- Да ли постоји истинитосна валуација a таква да сума тежина свих клауза које a задовољава буде бар W?

У приступу описаном у раду [52] решава се одговарајући MAXSAT. Дати алгоритам се састоји од низа фаза. У свакој фази, испитује се низ од неколико малих локалних измена у тренутној валуацији и бира се подниз са најбољим локалним побољшањем. Свака локална измена у низу се назива подфаза. Фаза има највише n подфаза (n је број исказних слова).

Нека је валуација на почетку фазе означена са a. На почетку i-те подфазе валуација је a_i, при чему је очигледно $a_1 = a$. За свако исказно слово x_j, израчунава се његов потенцијал p_{ij} за подфазу i. Овај потенцијал представља разлику у промени тежине задовољивих клауза у случају да x_j промени вредност (p_{ij} може имати и негативну вредност). Бирају се исказна слова са највећим потенцијалом која нису изабрана ни у једној од $i - 1$ претходних подфаза тренутне фазе. Валуација на почетку следеће фазе је a_{i+1} и добија се када се у валуацији a_i промени вредност изабраног исказног слова. Током рада алгоритма, такође, се прате промене у тежинама задовољивих клауза. Променљива Δ_i памти укупну промену тежине до подфазе i. Иницијално је $\Delta_0 = 0$ и $\Delta_{i+1} = \Delta_i + p_{ij}$ где је x_j исказно слово са највећом вредношћу p_{ij} у подфази i. Фаза се завршава или када се прође кроз свих n подфаза, или када након подфазе i вредност Δ_i постане непозитивна. Када се фаза прекине бирају се вредности из подфазе i^* за који је Δ_{i^*} највеће. Исказна слова са максималним потенцијалом у првој i^* подфази мењају вредности и почиње нова фаза. Алгоритам се завршава ако су у некој фази све вредности Δ_i непозитивне.

У раду [52] су представљени резултати добијени тестирањем две фамилије примера, које се разликују по начину генерисања вероватноћа. За прву фамилију су генерисани примери код којих параметар m (број калуза) добија вредности до 70, док за другу фамилију параметар m добија вредности до 80, а параметар n (број исказних слова), у оба случаја, добија вредности до 50. За примере малих димензија добија се, у оба случаја, велика успешност у решавању, за релативно мало време, док са порастом вредности параметра успешност опада до ситуације када решење не може да се нађе.

4.3.2 Табу претраживање

Табу претраживање (енг. *Tabu Search*, TS) је метахеуристика детаљно описана у књизи Гловера и Лагуне (енг. *Fred Glover* и *Manuel Laguna*)[30]. Основне идеје методе, као и анализа ефикасности и неки примери примене TS методе описани су у радовима [47, 95]. TS метода заснива процес претраживања на околини текућег минималног решења. Када се у току претраживања наиђе на локални минимум, потребно је дефинисати критеријум изласка из њега. Тај критеријум је код TS методе заснован на употреби меморије. За разлику од метода које чувају само тренутно најбоље решење и одговарајућу (тренутно минималну) вредност функције циља, TS метода памти кретање преко решења посећених у неколико претходних итерација. Те информације се користе за избор наредног решења као и за модификацију дефиниције околине у смислу избацивања неких „забрањених" решења. Обзиром на то јасно је да околина $\mathcal{N}(x)$ решења x варира од итерације до итерације, те се стога TS убраја у процедуре које се називају *технике динамичког претраживања околине*. Прецизније, извршавање основне верзије TS методе се састоји у следећем:

- ова метода систематски прати процес минимизације заснован на LS процедури све док не досегне локални минимум;
- затим се тај локални минимум искључује из околине за претраживање и тражи минимално решење у тако модификованој околини;
- искључена решења се памте у листи званој *табу листа* (TL) која представља забрањена решења у току неколико наредних итерација (дефинисаних дужином TL)
- по истеку забране, ова решења се враћају у околину, а нека друга постају забрањена.

Дужина табу листе може бити променљива, при чему би се у свакој итерацији генерисао случајан број који би представљао дужину TL за ту итерацију[15].

Критеријуми заустављања могу бити: максималан број итерација, максималан број итерација између две поправке најбољег решења, максимално дозвољено време извршавања и слично.

Суштина TS методе је у томе да се након достигнутог локалног минимума посећују решења која не воде побољшању вредности функције циља са идејом да се напусти околина тог локалног минимума и омогући претрага за новим. Да би се спречило поновно посећивање неког локалног минимума, тзв. циклирање, користи се табу листа (TL) у којој се чувају забрањена решења.

Спречавање циклирања није у потпуности загарантовано и зависи од дужине табу листе. Уколико је дужина листе превелика, диверсификација претраживања је већа и може довести до тзв. случајног кретања (енг. *random walk*). Са друге стране, уколико је дужина TL исувише мала, повећава се могућност циклирања.

Примена TS методе на PSAT

У раду [41] разматра се примена TS методе на решавање проблема код кога су полазне претпоставке исте као и код проблема решаваног LS методом: Посматра се скуп од m исказних формула $S_1, \ldots, S_m$ у којима се појављује n исказних слова $x_1, \ldots, x_n$ са уобичајеним Буловим операторима $\vee$, $\wedge$ и $\neg$. Вероватноћа да је свака од посматраних формула тачна је $\pi_1, \ldots, \pi_m$. Потребно је испитати да ли су дате вероватноће конзистентне са датим скупом формула.

Као и приликом примене LS методе на овакав проблем, дефинише се скуп свих истинитосних валуација w_j за $j = 1, \ldots, 2^n$ и потребно је одредити вероватносну расподелу $p_1, \ldots, p_{2^n}$ такву да је сума ових вероватноћа за сваку валуацију за коју је формула S_i тачна, једнака π_i за свако $i = 1, \ldots, m$. Овде се, такође, може дефинисати матрица $A = (a_{ij})$ димензије $m \times 2^n$ таква да је

$$a_{ij} = \begin{cases} 1 & \text{ако је } S_i \text{ тачно при валуацији } w_i \\ 0 & \text{у супротном} \end{cases}$$

у ком случају се проблем записује у облику

$$\begin{aligned} \mathbf{1}p &= 1 \\ Ap &= \pi \\ p &\geqslant 0 \end{aligned}$$

где је $\mathbf{1}$ јединични вектор врсте димензије 2^n, p и π вектори колоне $(p_1, \ldots, p_{2^n})^T$ и $(\pi_1, \ldots, \pi_m)^T$ респективно. Уколико се у скуп формула дода формула S_{m+1} са непознатом вероватноћом π_{m+1}, проблем се може дефинисати у терминима оптимизационих проблема. Вредност π_{m+1} није јединствена у случају да вектор $A_{m+1} = (a_{m+1,j})$ ($a_{m+1,j} = 1$ уколико је S_{m+1} тачно при валуацији w_j и $a_{m+1,j} = 0$ у супротном) представља линеарну комбинацију врста из матрице A. У супротном, дата дефиниција проблема имплицира границе вероватноће π_{m+1}. Сада дати проблем може да се запише у облику

$$\begin{aligned} &\min / \max \quad A_{m+1}p \\ &\text{у односу на:} \\ &\qquad\qquad \mathbf{1}p = 1 \\ &\qquad\qquad Ap = \pi \\ &\qquad\qquad p \geqslant 0 \end{aligned}$$

TS метода се, као и у случају примене LS методе, примењује на дуални проблем MAXSAT. Предложени алгоритам представља поједностављену верзију TS методе, означен је као SAMD (енг. *Steepest Descent Mildest Ascent*) и дат је псеудокодом 4.1 ([41]).

У раду [40] је посебно разматран избор параметара n_{rep} (број локалних промена) и ℓ (дужина Табу листе). Индекс j предствља правце промене, а параметар t_k означава преостали број итерација током којих су промене у правцу j забрањене. Када се

Алгоритам 4.1: TS алгоритам за PSAT

Селектовање иницијалног решења x_0;
$f_{opt} = f(x_0)$; $x_{opt} = x_0$;
$t_j = 0$ за $j = 1, \ldots, n$;
Repeat
  $f'_{opt} = f_{opt}$;
  $i = 0$;
  Repeat
    Изабрати $x_k \in \mathcal{N}(x_0)$ тако да $\delta_k = f(x_k) - f(x_0) = \min_{j|t_j=0} \delta_j$;
    $x_0 = x_k$;
    If $f(x_k) < f_{opt}$ **then** $f_{opt} = f(x_0)$; $x_{opt} = x_0$; **end if**
    If $\delta_k > 0$ **then** $t_k = \ell$;
    $t_j = t_j - 1$ за $t_j > 0$, $j = 1, \ldots, n$;
    $i = i + 1$;
  Until $i = n_{rep}$;
Until $f'_{opt} == f_{opt}$
x_{opt} је апроксимација оптималног решења

направи промена у правцу j, вредност параметра t_k се постави на ℓ. Што је већа вредност параметра ℓ, то је теже вратити се у околину локалног минимума. За тестирање методе коришћени су примери у којима је број исказних слова 100, а број клауза 500. За ове примере вариране су вредности параметара n_{rep} (узимане су вредности 100, 200, 500, 1000 и 2000) и ℓ (са вредностима 10, 15, 20,. . . , 40). Најбољи резулати су постизани за вредности параметра $n_{rep} = 1000$ и $\ell = 15$, при чему је показано да ова метода даје боље резулате од SA (енг. *Simulated Annealing*) методе.

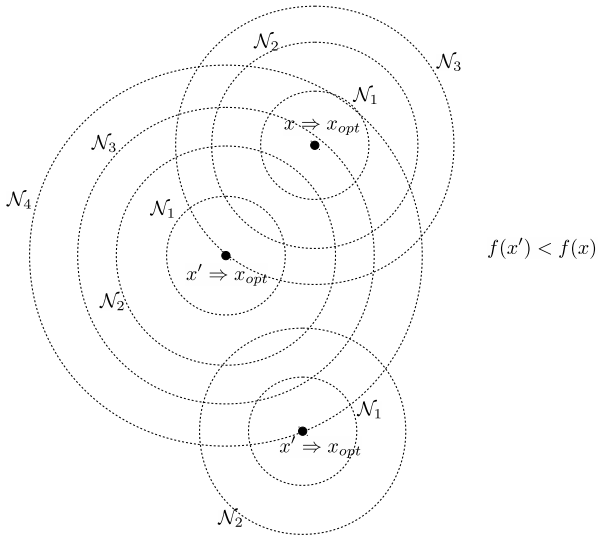
4.3.3 Метода променљивих околина

Методу променљивих околина (енг. *Variable Neighborhood Search*, VNS) предложили су Младеновић и Хансен (енг. *Pierre Hansen*) у свом раду [80]. Основна идеја методе је систематска промена околина унутар локалног претраживања. Неопходно је увести више околина, било да се мења метрика у односу на коју се дефинише околина, било да се повећава растојање у односу на исту метрику. Предложено је и неколико модификација, углавном намењених решавању проблема великих димензија [42, 43, 44, 45, 46]. VNS метахеуристика је заснована на три једноставне чињенице [46]:

1. локални минумум у односу на једну околину не мора бити и локални минимум у односу на неку другу околину;
2. глобални минимум је локални минимум у односу на све околине;
3. за већину проблема локални минимуми у односу на разне околине су међусобно блиски.

Прва чињеница оправдава увођење више околина. Друга чињеница не гарантује да решење које је локални минумум у односу на сваку од изабраних околина представља глобални минимум, већ да ако неко решење није локални минимум у односу на неку околину, онда сигурно није глобални минимум. Последица треће чињенице је потреба да се детаљније истражује најближа околина локалног минимума јер се ту очекује поправљање текућег најбољег решења[15].

Дефиниција 4.3.1. *Нека је $x \in X$ произвољно допустиво решење и нека је $\mathcal{N}_k$, $(k = 1, \ldots, k_{max})$ нека коначна колекција унапред дефинисаних околина. Тада је $\mathcal{N}_k(x)$ скуп решења у k-тој околини од x, тј. скуп решења која се у односу на усвојену метрику (број трансформација) налазе на растојању k од решења x.*



Слика 4.2: Употреба више околина у локалном претраживању

Слика 4.2 илуструје употребу више околина у решавању оптимизационих проблема. Многе метахеуристичке методе базиране на локалном претраживању користе једну, највише две околине.

Примена VNS алгоритма на проблем задовољивости у логици LPP_2^{ext}

У раду [23] је показана процедура свођења PSAT проблема на проблем линеарног програмирања, док је у раду [50] дат детаљан приказ примене VNS алгоритма на решавање добијеног система неједнакости. Описана процедура полази од формуле A, која представља конјункцију L тежинских формула логике LPP_2^{ext} у којој учествује n исказних слова, облика

$$A = \bigwedge_{i=1}^{L} a_1^j P^*(\alpha_1^j) + \cdots + a_k^j P^*(\alpha_k^j) \ \rho \ s^j$$

За формулу A се генерише одговарајући систем неједнакости чији је компактан облик

$$\begin{array}{l}\sum_{i=1}^{2^n} x_i = 1,\\ x_i \geqslant 0, i = \{1, \ldots, 2^n]\},\\ \sum_{i=1}^{2^n} c_{ij} x_i \ \rho \ s^j \qquad j \in \{1, \ldots, L\}\end{array}$$

где се за сваку тежинску формулу $a_1^j P^*(\alpha_1^j) + \cdots + a_k^j P^*(\alpha_k^j) \ \rho \ s^j$ која се јавља у A коефицијенти c_{ij} добијају груписањем по вероватноћама атома $c_{ij} = \sum_{l:at_i \in [\alpha_l^j]} a_l^j$. Према резултатима датим у радовима [23, 52] у вектору решења система (ако решење постоји) има највише $L+1$ ненула вредности. То значи да треба пронаћи $L+1$ атом из скупа свих атома $At(A)$ чије ће вероватноће бити веће од 0, при чему збир тих вероватноћа мора бити 1. Псеудокод који је коришћен је дат у наставку.

Алгоритам 4.2: VNS алгоритам за PSAT

```
x = initialSolution(); improve(x, heuristics)
while (not done())do
    k = 1
    while (k ⩽ k_max) do
        x' = shake(x, k); improve(x', heuristics); x'' = localSearch(x')
        if (x'' је боље од x) then
            x = x''; improve(x, heuristics); k ← 1
        else
            k = k + 1
        end if
    end while
improve(x, nelder-mead)
end while
```

Значења коришћених рутина су следећа:

- **initalSolution()** генерише иницијално решење тако што од случајно изабраних $10 \times (L+1)$ атома са додељеним вероватноћама $1/(L+1)$, бира $L+1$ атом за које је процењено да највише утичу на задовољивост неједнакости.
- **improve**(x,**heuristics**) коришћењем хеуристичких метода „поправља" вредности додељених вероватноћа, у циљу проналажења бољег решења.
- **shake**(x,k) има за циљ да у k-тој околини изабере нови атом у чијој околини ће направити ново локално претраживање.
- **localSearch**(x) претражује околину $\mathcal{N}_k$ тренутног решења тражећи боље решење. Због величине простора, вероватноће атома остају фиксиране, а процена нових решења се врши само на основу функције циља, која је у овом случају дефинисана формулом:

$$z(p) = \sum_{i=1}^{L} d_i(p) + g(p)$$

где је

$$d_j(p) = \begin{cases} \left(\sum_{i=1}^{L+1} c_{ij} p_i - s^j\right)^2 & \text{ако колона } j \text{ није задовољива} \\ 0 & \text{у супротном} \end{cases}$$

док функција $g(p)$ мери одступање вероватноћа од ограничења.

- **improve**(x,**nelder-mead**) поправља вредности вероватноћа користећи Нелдер-Мид метод (енг. *Nelder-Mead method*) чији је детаљан опис дат у поглављу А.2.

Предложена метода дала је добре резултате приликом тестирања случајно генерисаних задовољивих формула, чак и у случајевима са великим бројем тежинских формула и исказних слова.

4.3.4 Генетски алгоритам

Генетски алгоритам (енг. *Genetic Algorithm*, GA) представља метахеуристику која користи већи број решења (популацију) и базирана је на принципима природне еволуције. Методу је први предложио Холанд (енг. *John H. Holland*) у књизи [48], а детаљан опис методе дат је у књизи [32]. За разлику од локалног претраживања, GA врши модификацију решења по неким, унапред задатим, правилима или комбиновањем два или више решења у циљу генерисања нових решења. Овакав процес треба да симулира репродукцију живих организама у природи. Решења, која су чланови неке *популације* мешају се и производе *потомке* који би требало да задржавају добре особине својих *предака*. Основна идеја методе је да се изабере иницијална популација и да се затим кроз низ генерација еволуирања те популације врши поправка тренутно најбољег решења применом оператора. Основни оператори су *укрштање*, *мутација* и *селекција* [15]. Општи алгоритам ове методе је дат псеудокодом 4.3.

Алгоритам 4.3: Општи GA алгоритам

```
PopulationInit(N);
Repeat
    for(i = 0; i < N; i++) f_i = ObjectiveFunction();
// Креирање нове генерације
    Selection();
    Crossover();
    Mutation();
Until (задовољен критеријум заустављања);
```

У зависности од проблема који се решава, свака од коришћених функција се може имплементирати на више начина. Врло често се наведени оператори не примењују

на сам простор решења, већ се решења кодирају низовима симбола неког коначног алфабета. Најједноставније је бинарно кодирање (0 – 1 симболима), мада су у употреби и сложенија кодирања. Због једноставности, наведени оператори биће објашњени на примеру бинарног кодирања. Оваква репрезентација решења назива се *јединка* популације или *хромозом*, док се сваки од симбола назива *ген*.

PopulationInit(N) представља избор иницијалне популације која садржи N јединки. Често се овај скуп иницијалних решења бира на случајан начин.

ObjectiveFunction() се дефинише да би се за свако решење у посматраној популацији одредио „квалитет" израчунавањем функције циља (често се користи и назив *fitness*). Ова функција треба да представља нормализовано растојање конкретне јединке од најбоље јединке у популацији или од тренутно најбољег решења.

Selection() представља примену оператора селекције. На основу вредности функције циља различитим механизмима (рулет, турнир, униформно, елитистички) бирају се јединке које ће добити шансу да пређу у следећу генерацију, било непромењене, било кроз своје потомке.

Crossover() је примена оператора укрштања који се може дефинисати на различите начине. Решења изабрана оператором селекције, могу да се групишу у парове чијим ће се укрштањем добити два нова потомка, међутим, укрштање може да укључи и више јединки чиме се добија и више потомака. Уобичајена процедура укрштања је да се изаберу две јединке и да се на случајан начин изабере позиција у коду од које ће се извршити замена. На пример, ако је кôд дужине 8 бинарних симбола и 3 је позиција замене тада се укрштање може извести на следећи начин:

$$\begin{array}{ccc|ccccc} 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array} \quad \rightarrow \quad \begin{array}{cccccccc} 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{array}$$

Укрштање јединки може бити и на више позиција. Постоји велики број могућности начина избора две јединке за примену оператора укрштања. Често се јединке бирају на основу редоследа у популацији, а такође је могуће увести и изборе на основу примене оператора вероватноће, којим се процењује да ће дати пар јединки произвести ново најбоље решење.

Mutation() је оператор коме се подвргавају све јединке добијене након примене оператора укрштања. Циљ примене овог оператора је „промена" генетског материјала која треба да усмери потрагу ка неистраженим регионима. Најједноставнији облик мутације је да се на случајно изабраној позицији тренутни симбол замени новим симболом. На пример, мутација на позицији 5 у бинарном коду дужине 8 би дала следећи резултат:

$$1 \quad 0 \quad 1 \quad 1 \quad \underline{0} \quad 0 \quad 1 \quad 0 \quad \rightarrow \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0$$

Након завршетка ових операција добија се нових N јединки које заједно са јединкама текуће популације конкуришу за улазак у следећу генерацију.

Критеријум за заустављање може бити максималан број генерација, број генерација без поправке тренутно најбољег решења, максимално дозвољено време извршавања и слично.

Примена GA методе на PSAT у логици LPP_2^{ext}

У радовима [85, 86] је детаљно објашњен поступак примене GA методе на PSAT. Као и приликом примене VNS методе на овај проблем, посматрана је формула A од L тежинских формула у којима се јавља k исказних слова. Трансформација на одговарајући систем се врши на исти начин. При решавању проблема, дефинише се популација јединки, при чему се свака јединка састоји од L парова облика (атом, вероватноћа) чиме се описује вероватносни модел. За оцењивање јединки коришћене су две различите функције. Прва функција t представља укупан број тежинских формула из A које су тачне за јединку M, што значи да ако је $t(M)$ једнако L тада је M решење. Друга функција d представља меру незадовољивости јединке M и има облик

$$d(M) = \sqrt{\sum_{j=1}^{L} \left(\sum_{i=1}^{2^k} c_{ij} x_i - s^j \right)^2}$$

Циљ је функцију d свести на вредност 0.

Популација у описаном методу се састоји од 10 јединки, селекција се врши рангирањем, укрштање је у једној позицији и примењена је једноставна метода мутација са вероватноћом 0.03. У методу је додата и једноставна хеуристика која након, селекције, за јединке поправља тренутне вероватноће атома.

У раду [86] су приказани резултати тестирања који су обухватили формуле различитих величина, прецизније, број исказних слова је вариран од 50 до 200, а број тежинских формула од 50 до 1000. За изабране примере коришћене су различите варијанте GA методе код којих су се разликовали опертори мутације, вероватноће мутације, а неке су укључивале и процедуре локалне претраге. У резултатима је приказано да успешност у решавању расте са повећањем вероватноће мутације, међутим код малих примера успешност, након одређене вредности вероватноће мутације, почиње да опада, пошто процедура почиње да се понаша као процедура случајне претраге. Такође је код већих примера показано да мутација атома има већи утицај од мутације вероватноће атома. Додавање процедуре локалног претраживања у алгоритам је повећало успешност у проналажењу решења.

4.3.5 Комбиновање метода, хибридизација

Већ је речено да ниједна метахеуристичка метода не гарантује оптималност добијеног решења. Чак је и процена квалитета решења изузетно тешка. Разним методама поређења, показано је да не постоји хеуристика која је једнако успешна за решавање

сваког проблема, па чак и за различите варијанте истог проблема може се показати да су неке хеуристике успешније од других.

Да би се повећала ефикасност метахеуристичких метода, све чешће се прибегава комбиновању, хибридизацији, две или чак и више метахеуристичких метода. У литератури се често могу наћи комбинације GA и TS, или GA и LS, или VNS и TS и слично. Осим тога могу се наћи и комбинације метахеуристичких и егзактних метода [15].

У раду [87] приказана је комбинација GA и VND (скраћено од *Variable Neighborhood Descent*) за решавање проблема задовољивости формула LPP_2^{ext} логике. Тестирања која су приказана у раду садрже меру успешности у доказивању задовољивости и времена извршавања програма под истим условима. Добијени резулати су показали велику супериорност хибрида GA-VND у односу на примену само GA методе.

4.4 Метода оптимизације колонијом пчела

Оптимизација колонијом пчела (*Bee Colony Optimization*, BCO) представља новију метахеуристичку методу инспирисану природом, прецизније понашањем пчела у потрази за храном ([17, 72, 73, 74, 118]). Ова метахеуристичка метода припада класи метода инспирисаних интелигенцијом групе (роја, колоније, енг. *Swarm Intelligence*). Овој класи метода такође припадају и алгоритми инспирисани мравима (*Ant Colony Optimization*), бактеријама (*Bacterial Colony Optimization*), сивим вуковима (*Grey Wolf Optimizer*), слепим мишевима (*Bat Algorithm*), имуним системом (*Artificial Immune Systems*), светлећим црвима (*Glowworm Swarm Optimization*) и многи други [11, 121, 122, 123]. Основна идеја BCO методе је да се искористи сличност између начина на који пчеле у природи траже храну [10] и начина на који оптимизациони алгоритми траже оптимално решење оптимизационог проблема [116]. Осим основне конструкционе BCO методе, развијена је варијанта са поправком BCOi (BCO *improvement*) која уместо конструисања решења, врши поправку постојећих решења [16]. Осим при решавању разних комбинаторних проблема, BCO метода се показала врло поузданом при решавању различитих нестандардних проблема, попут проблема који садрже непрецизне податке [78, 117] или укључују више оптимизационих критеријума [120]. Ипак она раније није коришћена за решавање проблема задовољивости. У наставку ће бити описано понашање пчела у природи, које је послужило као инспирација за развој методе, као и општи BCO алгоритам који је коришћен за решавање широке класе проблема.

4.4.1 Пчеле у природи

Пчеле у природи траже храну претражујући поља у околини њихове кошнице. Оне сакупљају и складиште храну коју касније користе и друге пчеле. Најчешће,

у почетном кораку сакупљања хране, пчеле-истраживачи претражују околину кошнице. Након завршене претраге пчеле-истраживачи се враћају у кошницу и информишу остале пчеле у кошници о локацији, количини и квалитету расположиве хране у области коју су претражиле. У случају да су пронашле нектар у истраженој области, пчеле-истраживачи „плешу" у кошници и тиме „рекламирају" локацију са храном охрабрујући остале чланице колоније да их прате ([10]).

Уколико пчеле одлуче да напусте кошницу и сакупљају нектар, оне ће пратити пчеле-истраживаче које су „плесале", до откривене локације са храном. Након сакупљања нектара, пчеле се враћају у кошницу и смештају сакупљену храну. Након тога је могуће неколико сценарија:

1. пчела може да напусти локацију са храном и поново постаје неопредељани следбеник;
2. пчела може да настави са прикупљањем хране са претходне локације, при чему неће „регурутовати" остатак колоније;
3. пчела може „плесом" да регрутује друге пчеле из кошнице пре него што се врати прикупљању хране.

Пчела усваја једну од наведених могућности, а уколико више пчела покушава истовремено да регрутује пчеле из кошнице својим плесом, тада није јасно како пчеле одлучују коју ће „плешућу" пчелу да прате, мада се може закључити да избор зависи од квалитета извора хране [10]. Описани процес се наставља све док се у кошници сакупља нектар и док се истражују нове области са потенцијалним извором хране. Ово су кључни аргументи за развој BCO алгоритма.

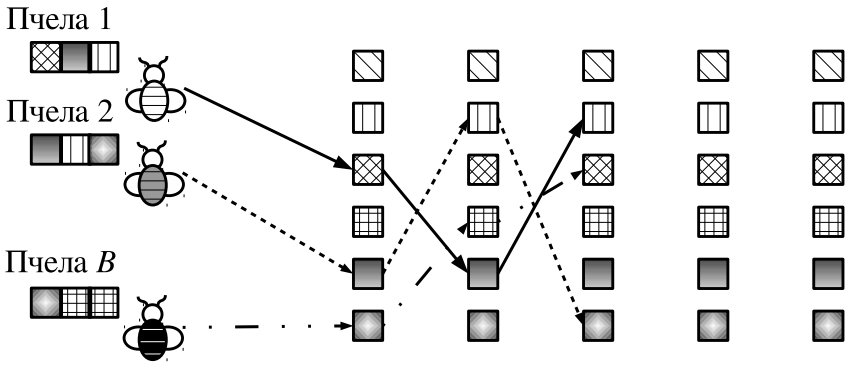
4.4.2 BCO алгоритам

Лучић и Теодоровић су у својим радовима [72, 73, 74] први који су користили принципе колективне интелигенције пчела за решавање комбинаторних оптимизационих проблема. Основна идеја је да се направи систем мулти-агената (колонија *вештачких пчела*) који ће тражити добро решење проблема, користећи принципе које користе пчеле приликом сакупљања нектара. Колонија вештачких пчела је често веома мала (на пример 5 до 10 пчела). Током потраге за решењем пчеле сарађују и размењују информације у циљу проналажења бољег решења. Коришћењем колективног знања и разменом информација, пчеле се концентришу на области које више обећавају. Корак по корак пчеле граде и/или поправљају своја решења. BCO потрага се обавља по итерацијама док се не испуни неки од унапред дефинисаних услова за заустављање, који су већ поменути приликом описа других метахеуристичких метода.

Популација вештачких пчела се састоји од B индивидуа које у сарадњи траже оптимално решење. Свака вештачка пчела је одговорна за једно решење проблема.

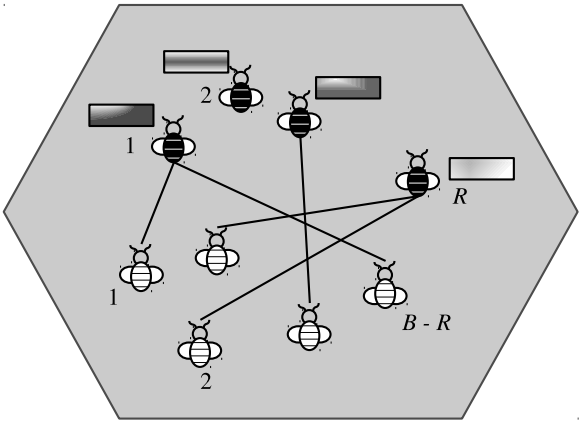
Алгоритам се састоји од две фазе које се смењују лет унапред/лет уназад (енг. *forward pass/backward pass*) формирајући један корак алгоритма. Током лета унапред пчела истражује простор решења и примењује унапред дефинисане кораке којима гради и/или поправља решење. Лет уназад је фаза током које се обавља размена информација међу пчелама. Лет унапред и лет уназад се током једне итерације смењују NC пута, а вредност параметра NC зависи од карактеристика проблема. Уколико је вредност параметра NC мала, пчеле ретко комуницирају. Са повећањем вредности параметра NC пчеле учесталије размењују информације о квалитету решења.

Уколико у изградњи решења учествује B пчела означених са: Пчела 1, Пчела 2, до Пчела B, које граде решење од n компоненти и ако је $NC = n$, тада оне у сваком лету унапред могу свом решењу да додају по једну компоненту. На слици 4.3 је приказана једна од могућих изградњи тих решења након 3 лета унапред уколико не долази до регрутације међу тим пчелама.



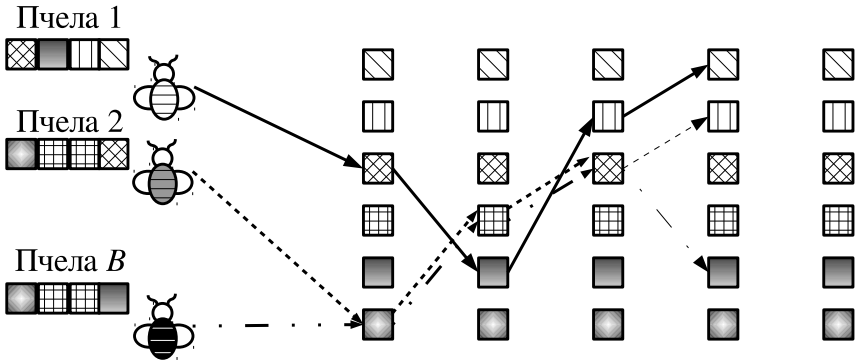
Слика 4.3: Изградња решења након 3 лета унапред

Након добијања новог (парцијалног) решења, пчеле размењују информације - фаза лет уназад. У природи пчеле се враћају у кошницу и ритуалом плеса обавештавају остале о храни коју су нашле. У алгоритму, лет уназад почиње оценом квалитета решења сваке пчеле, тј. рачуна се вредност функције циља за тренутно решење. Након тога, свака пчела одлучује, са одређеном вероватноћом, да ли ће остати верна (лојална) свом решењу или не. При овом одлучивању, пчеле са бољим решењем имају већу шансу да остану верне свом решењу. Насупрот пчелама у природи, вештачке пчеле које су остале верне свом решењу, уједно врше и „регрутацију", тј. њихово решење разматрају и друге пчеле. Када пчела напусти своје решење, она постаје неопредељена и мора да одабере једно од рекламираних решења. Ове одлуке се, такође, доносе са одређеном вероватноћом, при чему боља решења имају већу шансу да буду изабрана. Овиме се у сваком лету уназад пчеле деле на две групе, као што је показано на слици 4.4, где је R пчела остало верно свом решењу и врши регрутацију, а $B - R$ пчела је неопредељено. Вредности R и $B - R$ се могу мењати од једног лета уназад до другог.



Слика 4.4: Регрутација неопредељених пчела

Након што неопредељена пчела преузме решење друге пчеле, у следећем лету унапред она наставља да гради преузето решење, независно од пчеле чије је решење преузето. Ако у ситуацији са слике 4.3, Пчела 2 преузме решење Пчеле B, у следећем лету унапред наставља градњу тог решења и добија се ситуација као на слици 4.5.



Слика 4.5: Наставак изградње решења након преузимања решења друге пчеле

Две фазе алгоритма претраге, лет унапред и лет уназад, се смењују у циљу добијања решења (по једно решење за сваку пчелу). Након NC смена одређује се најбоље решење и тиме се завршава једна итерација BCO (BCOi) алгоритма. Након овога сва решења се бришу и почиње нова итерација. BCO (BCOi) алгоритам се извршава итерацију по итерацију док се не испуне критеријуми за заустављање (максималан број итерација, максималан број итерација без поправљања решења, максимално дозвољено време извршавања и слично). На крају се најбоље решење враћа као коначан резултат алгоритма.

Главна предност BCO (BCOi) алгоритма је мали број параметара који се подешавају пре покретања алгоритма:

- B - Број пчела које учествују у решавању проблема;

- NC - Број летова унапред (уназад) у једној итерацији.

Једноставност псеудокода BCO (BCOi) алгоритма је такође велика предност. Псеудокод је приказан алгоритмом 4.4.

Алгоритам 4.4: BCO алгоритам

Do
Иницијализација (празног) решења за сваку пчелу;
For $(s = 0; s < NC; s++)$
//Лет унапред
For $(b = 0; b < B; b++)$
Израчунавање могућих корака;
Случајан избор и примена једног корака;
//Лет уназад
For $(b = 0; b < B; b++)$
Оцена (делимичног/комплетног) решења за пчелу b;
For $(b = 0; b < B; b++)$
Одлука о лојалности пчеле b тренутном решењу;
For $(b = 0; b < B; b++)$
If (b је неопредељена) случајан избор регрутера;
Оцена свих решења и проналажење најбољег;
While (критеријум за заустављање није испуњен);

Детаљан опис примене BCOi алгоритма за доказивање задовољивости формула логике $LPCP$ биће дат у поглављу 5, док је дифолтно закључивање применом овог алгоритма дато у поглављу 6.

Глава 5

Примена методе оптимизације колонијом пчела у вероватносним логикама

У овом поглављу ће бити детаљно описан начин имплементације BCOi методе за решавање проблема задовољивости формула логике $LPCP$, тј. за проблем CPSAT-ε. Поред тога, биће приказани резултати добијени тестирњем, као и поређење добијених резултата са резултатима добијеним применом Фуније-Моцкин методе на проблем CPSAT-ε.

5.1 Примена BCOi алгоритма на проблем CPSAT-ε

У делу 2.3 је описан механизам за свођење проблема задовољивости дате конјункције вероватносних формула A на систем линарних неједнакости. Овим се проблем CPSAT-ε своди на проблем линеарног програмирања над вероватноћама. На основу резултата, објављених у радовима [23, 28, 50], број атома са вероватноћама већим од 0 потребних да гарантују решење добијеног система (уколико решење постоји) је $L+1$, где је L број неједнакости у добијеном систему неједнакости. У добијеном систему се не налази услов да је сума вероватноћа свих атома 1, али се он подразумева, па је из тог разлога потребно да се вектору решења нађе $L+1$ атом, а не L. Дакле, решење система се може представити вектором од $L+1$ атома

$$a = [a_1, a_2, \ldots, a_{L+1}]$$

где је a_i, $i = 1, \ldots, L+1$ атом из скупа свих атома $At(A)$, са додељеним вероватноћама

$$x = [x_1, x_2, \ldots, x_{L+1}]\,.$$

Вероватноће атома који се не налазе у вектору a су 0. Ако је познат вектор $a = [a_1, a_2, \ldots, a_{L+1}]$ који представља решење система, тада се систем може записати у

скраћеном облику

$$\sum_{j=1}^{L+1} c_{ij}x_i \ \rho_i \ 0, \qquad i = 1, \dots L \tag{5.1.1}$$

где су c_{ij} $(i = 1, \dots, L, j = 1, \dots L+1)$ коефицијенти добијени груписањем уз вероватноћу x_i атома a_i и $\rho_i \in \{\leqslant, <, \geqslant, >\}$ $(i = 1, \dots, L)$.

Имплементирани BCOi алгоритам се састоји од четири фазе. Прва фаза је на почетку сваке итерације и она представља генерисање иницијалног решења система. Током друге фазе врши се модификација решења. Фазе три и четири чине лет уназад, при чему се у фази три врши поређење решења, док се у фази четири обавља регрутација. У наставку ће свака од фаза бити детаљно објашњена.

5.1.1 Иницијално решење

Свака итерација BCOi алгоритма почиње генерисањем и оценом иницијалног решења за сваку пчелу. Најпре, свака пчела на случајан начин генерише $5 \times (L+1)$ различитих атома. Сваком атому се додељује иста вероватноћа $1/(L+1)$ и оцена атома која представља меру утицаја тог атома на задовољивост неједнакости.

Атому a_k, $(k \in \{1, \dots L+1\})$ одговара колона $c_k = [c_{1k}, c_{2k}, \dots, c_{Lk}]^T$ система (5.1.1). Ако је коефицијент c_{ik} позитиван и налази се у неједнакости у којој је симбол релације $\geqslant$ или $>$, он се може искористити да „погура " ка задовољивости те неједнакости. У овом случају се вредност тог коефицијента додаје на оцену одговарајућег атома. У случају да је симбол релације $<$ или $\leqslant$ атом не доприноси задовољивости те неједнакости, па се вредност коефицијента одузима од оцене. Слична ситуација је и када је коефицијент негативан. На основу овога, оцена атома a_k се може рачунати по формули

$$grade(a_k) = \sum_{i=1}^{L} c_{ik} \cdot \mathrm{sgn}(i),$$

где је $\mathrm{sgn}(i)$ знак i-те неједнакости система

$$\mathrm{sgn}(i) = \begin{cases} 1, & \text{ако је } \rho \in \{\geqslant, >\}, \\ -1, & \text{ако је } \rho \in \{\leqslant, <\}. \end{cases}$$

За сваку пчелу формира се иницијално решење тако што се од $5 \times (L+1)$ генерисаних атома бира $(L+1)$ атома са најбољим оценама.

5.1.2 Модификација решења

Основни корак BCOi методе је модификација решења која се врши у сваком од NC летова унапред у оквиру једне итерације. За оцену вероватносне расподеле користи се функција циља. Функција циља се у овом случају дефинише као растојање

леве и десне стране неједнакости система 5.1.1. Нека је a тренутно решење, функција циља z се дефинише као

$$z(x) = \sum_{i=1}^{L} d_i(x), \tag{5.1.2}$$

где је d_i, растојање између леве и десне (вредности десних страна неједнакости у систему су 0) i-те неједнакости, дефинисано са

$$d_i(x) = \begin{cases} (\sum_{j=1}^{L+1} c_{ij}x_j)^2 & \text{ако } i\text{-та неједнакост није задовољена,} \\ 0 & \text{у супротном.} \end{cases} \tag{5.1.3}$$

Вредност функције циља је ненегативна и циљ је да постане 0, уз услов ограничења вероватноћа (вредности x_i су ненегативне и њихова сума је 1). Ипак, није довољно да вредност функције циља добије вредност 0, због неједнакости у којима је знак релације $<$ или $>$, а вредност леве стране је 0, па те неједнакости нису задовољене.

Уколико решење није пронађено користи се хеуристички приступ за поправљање вероватноћа изабраних атома, слично као у раду [50]. Ова хеуристичка оптимизација се састоји од две независне хеуристике назване најгора незадовољива пројекција (енг. *Worst Unsatisfied Projection*, WUP) и похлепно пребацивање (енг. *Greedy Giveaway*, GG).

Најгора незадовољива пројекција

Ова хеуристика покушава да поправи неједнакости система (5.1.1) које су највише незадовољене. Из система се бира пет неједнакости које имају највећу вредност $d_i(x)$. Овим неједнакостима одговара хипер-раван која ограничава простор решења. У покушају да се достигне простор решења, одређују се пројекције вероватноћа на ову хипер-раван након чега се добијене вредности нормализују. Пројекција тачке $x = [x_1, x_2, \ldots, x_{L+1}]$ на хипер-раван коју дефинише i-та неједнакост, а која се описује једначином

$$\sum_{j=1}^{L+1} c_{ij}x_j = 0$$

се добија формулом

$$x'_j = x_j - \frac{\sum_{k=1}^{L+1} c_{ik}x_k}{\sum_{k=1}^{L+1}(c_{ik})^2}$$

Коришћењем ове формуле, вероватноће се за сваку изабрану неједнакост померају ка задовољивости у правцу нормалном на хипер-раван. Овај поступак се понавља највише 5 пута, а све док се решење поправља, при чему се сваки пут бира 5 „најгорих" неједанкости.

Похлепно пребацивање

Систем (5.1.1) је врло „редак", тј. велики проценат коефицијената у систему је нула, што омогућава решавање система „ручно".

Врши се избор најгоре незадовољене неједнакости (означене са i) и најбоље задовољене неједнакости (означене са j). У овим неједнакостима се посматрају парови коефицијената уз исте атоме, такве да је у једној неједнакости коефицијент који има вредност нула, а у другој коефицијент који има не-нула вредност и обрнуто. На пример k_1 и k_2 су парови таквих коефицијената у две изабране неједнакости које имају облик

$$\begin{array}{cccccccccccc} \ldots & + & c_{ik_1} & \cdot & x_{k_1} & + \ldots + & 0 & \cdot & x_{k_2} & + \ldots & \rho_i & 0 \\ \ldots & + & 0 & \cdot & x_{k_1} & + \ldots + & c_{jk_2} & \cdot & x_{k_2} & + \ldots & \rho_j & 0 \end{array}$$

Сада се вероватноћа p_{k_1} у i-тој врсти може променити тако да допринесе задовољивости ове врсте, док се вероватноћа p_{k_2} може променити у супротном смеру, што ће смањити задовољивост j-те врсте. Ово пребацивање вероватноћа се може поновити за све овакве парове у одабраним неједнакостима. Обзиром да је систем редак, ове промене вероватноћа не би требало да много поремете задовољивост осталих неједнакости.

Редослед извршавања рутина у оквиру BCOi алгоритма

Комбинација ове две хеуристике, најпре примена WUP, а затим GG, доводи до значајног поправљања вредности функције циља (5.1.2). Оне се примењују први пут непосредно након генерисања иницијалног решења, а затим у сваком лету унапред. Сваки лет унапред почиње додавањем нових $L/5$ атома, са додељеном вероватноћом $1/(L{+}1)$, у скуп атома који чине тренутно решење, а потом се задржава у решењу $L{+}1$ са најбољом оценом. На крају се примењују две описане хеуристике за поправљање решења.

5.1.3 Поређење решења

Оцена и поређење решења додељених пчелама се обавља након модификације решења и то представља почетак фазе лет уназад. Решења се оцењују на основу функције циља (5.1.2).

Нека је са $z_b (b = 1, 2, \ldots, B)$ означена вредност функције циља за пчелу b. Нормализоване вредности функције циља z_b биће означене са O_b, при чему се нормализација обавља по формули

$$O_b = \frac{z_{max} - z_b}{z_{max} - z_{min}}, \quad O_b \in [0, 1], \quad b = 1, 2, \ldots, B \tag{5.1.4}$$

где су z_{min} и z_{max} најмања и највећа вредност функције циља, респективно, међу вредностима функције циља решења које генеришу све пчеле.

Вероватноћа да ће пчела b (на почетку новог лета унапред) остати верна свом претходном решењу не зависи од проблема који се решава. Ова вероватноћа се одређује на исти начин као и у раду [16], по формули

$$p_b = e^{-(O_{max}-O_b)/u}, \quad b = 1, 2, \ldots, B, \tag{5.1.5}$$

где је u редни број лета унапред (добија вредности $1, 2, \ldots, NC$) и O_{max} је највећа вредност међу свим O_b вредностима. Свака пчела, користећи релацију (5.1.5) и генератор случајних бројева, доноси одлуку да ли ће наставити са развојем досадашњег решења или ће постати неопредељени следбеник.

5.1.4 Регрутација

Вероватноћа да ће решење пчеле регрутера b бити изабрано од стране неке пчеле која је неопредељени следбеник, се израчунава по формули [16]:

$$p_b = \frac{O_b}{\sum_{k=1}^{R} O_k}, \qquad b = 1, 2, \ldots, R \tag{5.1.6}$$

где је O_k нормализована вредност функције циља k-тог рекламираног решења, док R означава укупан број регрутера. Сваки неопредељени следбеник бира једног регрутера, на основу релације (5.1.6) и генератора случајних бројева. Ово заправо значи да се решење регрутера копира у решење следбеника. Од овог тренутка па на даље, обе пчеле, независно, претражују простор решења и исто решење модификују на различите начине.

Једна итерација се зауставља када се пронађе решење (тј. када су све неједнакости система задовољене) или када се фазе лет унапред/лет уназад изврше NC пута. Уколико решење није пронађено, структуре података са решењима се бришу и почиње нова итерација. BCOi алгоритам се извршава итерацију за итерацијом док се не стекне услов за заустављање, што у овом случају значи да или је пронађено решење система или је достигнут максималан број итерација. Псеудокод BCOi алгоритма, дат у поглављу 4.4, се сада може записати детаљније псеудокодом 5.1.

У наставку ће бити приказани резултати добијени применом описаног алгоритма на случајно генерисане задовољиве формуле.

5.2 Анализа и поређење резултата добијених тестирањем

За потребе тестирања генерисан је скуп од 57 задовољивих формула (све датотеке са тестираним фомулама се могу наћи на адреси
http://imi.pmf.kg.ac.rs/tstojanovic/publications). Ознаке N и M су коришћене за

Алгоритам 5.1: BCOi алгоритам за CPSAT-ε

```
BCOi(B, NC)
  Iter = 1; found = true;
  Do
   // Генерисање колоније
    For (b = 0; b < B; b++ )
      Случајно генерисање 5 × (L + 1) атома са вероватноћама 1/(L + 1);
      Оцена свих атома и избор L + 1 најбољих;
      Поправљање решења коришћењем WUP и GG хеуристика;
      If (решење пронађено) return found;
    For (s = 0; s < NC; s++)
   //Лет унапред
      For (b = 0; b < B; b++)
        Додавање нових L/5 атома са вероватноћама 1/(L + 1);
        Оцена свих атома и избор L + 1 најбољих;
        Поправљање решења коришћењем WUP и GG хеуристика;
        If (решење пронађено) return found;
   //Лет уназад
      For (b = 0; b < B; b++)
        Оцена решења за пчелу b коришћењем (5.1.4);
      For (b = 0; b < B; b++)
        Одлука о лојалности решењу за пчелу b коришћењем (5.1.5);
      For (b = 0; b < B; b++)
        If (b је неопредељена) случајан избор регрутера коришћењем (5.1.6);
    Iter++;
  While (Iter ⩽ Iter_max);
  return not(found);
```

означавање укупног броја исказних слова и броја CP формула, респективно, у формули чији је облик дат у поглављу 2.3, а која се може записати и у облику

$$\bigwedge_{i=1,M} CP_{\rho_i t_i}(\alpha_i, \beta_i), \qquad t_i \in \begin{cases} \mathbb{Q}[\varepsilon] \cap [0,1], & \rho_i \in \{\geqslant, >, \leqslant, <\} \\ \mathbb{Q}[0,1], & \rho_i = \approx \end{cases} \tag{5.2.1}$$

Свака исказна формула која се појављује као подформула CP формуле је у дисјунктивној нормалној форми и састоји се од највише M конјункција, при чему се свака конјункција састоји од највише $N/3$ литерала. Под литералом се сматра исказно слово или његова негација.

За потребе поређења резултата тестирања развијен је и аутоматски доказивач за CPSAT-ε базиран на Фурије-Моцкин методи елиминацаије (FME). Сам метод је детаљно објашњен у делу A.1. Да би поређење BCOi и FME методе за решавање CPSAT-ε проблема било могуће, генерисани су примери са малим бројем исказних слова и малим бројем CP формула. Примери су генерисани за следеће (N, M) комбинације: $(3,3)$, $(3,4)$, $(4,3)$, $(4,4)$, $(4,5)$, $(5,5)$, при чему су за сваку комбинацију

генерисана по 3 различита примера.

Све вероватноће и константе у систему (5.1.1) су из Хардијевог поља $Q[\varepsilon]$. То значи да су за потребе овог тестирања представљени као полиномне функције које зависе од ε са реалним коефицијентима двоструке прецизности, тј.

$$\eta = h_0 + \sum_{j=1}^{k} h_j \varepsilon^j$$

Овакав начин представљања довео је до проблема дељења нулом приликом примене FME методе. Из тог разлога, за потребе примене FME методе коришћено је и представљање рационалном функцијом која зависи од ε, са реалним коефицијентима двоструке тачности, тј.

$$\eta = \frac{\sum_{j=-k}^{k} h_j \varepsilon^j}{1 + \sum_{j=1}^{k} h_j \varepsilon^j}$$

Код оба начина представљања максимална вредност параметра k је ограничена на 4 (користећи исту идеју као и у раду [93]). При томе су коефицијенти који се јављају уз $\varepsilon^5, \varepsilon^6, \ldots$ игнорисани и нису задржавани у запису, као превише мали да би утицали на резултат. Приликом примене BCOi методе увек је коришћено представљање полиномном функцијом, пошто се ту проблем дељења нулом јавља ретко и из другачијег разлога, а решава се заустављањем текуће итерације и преласком на нову итерацију.

Све имплементације тестиране су на кластеру који се састоји од чворова са 2 AMD Opteron 2.1GHz 6272 процесора са 48 GB RAM са кернелом верзије 2.6.32 x64, gcc компајлером верзије 4.4.3.

У табели 5.1 приказани су резултати тестирања малих случајно генерисаних задовољивих формула.

Колоне Полином и Рационал у табели 5.1 садрже времена добијена применом FME методе са полиномним и рационалним, респективно, представљањем бројева из Хардијевог поља $Q[\varepsilon]$. Колона BCOi садржи просечне вредности добијене у 10 независних извршавања BCOi методе са вредностима параметара $NC = 30$ и $B = 10$ и са максимално 50 итерација. За сва поља у колони BCOi у којима су дата времена, значи да је свако од 10 извршавања било успешно. Поља која садрже симбол „–" означавају да BCOi метода није успела да нађе решење проблема за дефинисане параметре. Приликом примене FME методе број неједнакости у систему експоненцијално расте током елиминације непознатих из система. Да би FME метод могао коректно да се примени за формулу која има N исказних слова и M CP формула, систем мора да садржи $2^N + 2$ неједнакости којима се ограничавају вредности вероватноћа и за сваку CP формулу по 2 или 3 неједнакости у зависности да ли је симбол релације $\approx$ или не. То значи да се формула која садржи 5 CP формула и 4 исказна слова трансформише у систем који има 44 до 49 неједнакости и 16 непознатих. У примеру $N = 4$, $M = 5$, *Пример* $= 1$ након елиминације 9 непознатих из система

Табела 5.1: Резултати тестирања малих формула применом FME и BCOi методе

N, M, Пример	Протекло време(s)		
	Полином	Рационал.	BCOi
3, 3, 0	0.0000	0.0000	0.0010
3, 3, 1	DIV 0	1.7600	–
3, 3, 2	0.0000	0.0000	0.0010
3, 4, 0	0.0600	0.5600	0.0000
3, 4, 1	0.0000	0.0300	0.0010
3, 4, 2	DIV 0	683.580	–
4, 3, 0	0.2800	1.1100	0.0020
4, 3, 1	26.1900	88.7700	0.0010
4, 3, 2	166.5300	2116.7100	0.0020
4, 4, 0	0.0000	0.0000	0.0020
4, 4, 1	DIV 0	0.3400	–
4, 4, 2	0.1700	0.6800	0.0020
4, 5, 0	MEM	MEM	0.0020
4, 5, 1	MEM	MEM	0.0020
4, 5, 2	MEM	MEM	0.0040
5, 5, 0	MEM	MEM	0.0280
5, 5, 1	0.0200	0.0000	0.0030
5, 5, 2	MEM	MEM	0.0050

добија се систем који има више од $1.5 \cdot 10^8$ неједнакости, што захтева више меморијског простора него што је расположиво у RAM меморији, па тако метод не може да заврши израчунавање. У табели 5.1 проблем недовољног RAMa је обележен са MEM, док је проблем дељења нулом обележен са DIV 0.

На основу резултата приказаних у табели 5.1, може се закључити да се променом начина представљања бројева из Хардијевог поља $Q[\varepsilon]$ битно повећава време извршавања применом FME методе. Наравно, промена начина представљања бројева решава проблем дељења нулом, али проблем са недостатком меморије остаје присутан. Са друге стране, време извршавања BCOi методе, код већине примера, битно је мање него код FME методе и не повећава се битно са повећањем параметара N и M. Приметно је да примери код којих се јавио проблем дељења нулом не могу да се реше ни BCOi методом, пошто се ради о малим примерима код којих се скуп атома који се налазе у решењу мало мења, па се тражење решења система своди на модификацију вредности вероватноћа. У овом случају, слично као код примене FME методе где се проблем дељења нулом јавља због блиских вредности коефицијената у неједнакостима, и код примене WUP хеуристике лако може доћи до истог проблема, што онемогућава проналажење решења.

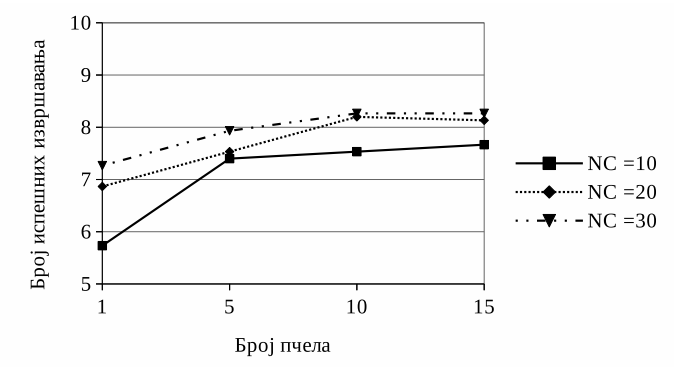
Приликом почетних тестирања варирани су параметри хеуристика WUP и GG, при чему се пошло од вредности коришћених у раду [50], а на крају су усвојене вредности дате у опису самих хеуристика у претходном поглављу. Приликом првих тестирања, такође, је уочено да приликом сваког стартовања у коме је решење пронађено, број извршених итерација је битно мањи од 50, и да даље повећање броја итерација не доприноси успешности. Из тог разлога је максималан број итерација ограничен на 50. Да би се одредила најуспешнија комбинација параметара B и NC за BCOi методу, из скупа свих генерисаних примера, издвојен је мањи број примера

различитих димензија. За овај скуп примера, вариране су вредности параметара B и NC, док су параметри хеуристика WUP и GG и максималан број итерација остали фиксирани.

Сваки пример је тестиран по 10 пута са истим вредностима параметара. Изабран је по један пример за следеће (N, M) комбинације: $(3, 3)$, $(5, 5)$, $(10, 30)$, $(10, 50)$, $(10, 100)$, $(20, 20)$, $(20, 50)$, $(20, 100)$, $(20, 250)$, $(30, 30)$, $(30, 60)$, $(30, 100)$, $(30, 250)$, $(40, 40)$, $(50, 50)$. Приликом тестирања мерени су: просечан број успешних стартовања, просечно време извршавања успешних итерација, просечан број израчунавања функције циља и просек укупног времена извршавања сваког стартовања (успешног и неуспешног). Добијени резултати су приказани на сликама 5.1, 5.2, 5.3, 5.4 и у табели 5.2.

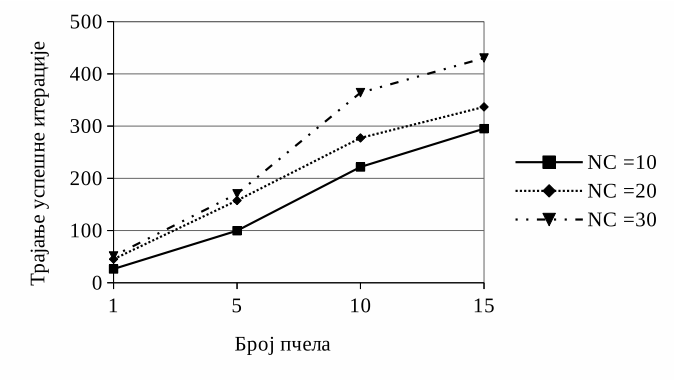
Табела 5.2: Резултати добијени варирањем параметара B и NC

NC	B	Просечан број успешних стартовања у 10 покушаја	Просечно време извршавања успешне итерације ()s	Просечан број израчунавања функције циља	Просек укупног времена извршавања сваког стратовања(s)
10	1	5.73	26.43	243.65	578.18
	5	7.40	99.65	845.43	1945.60
	10	7.53	221.79	1480.60	2584.07
	15	7.67	295.08	2115.50	2782.54
20	1	6.87	45.21	399.99	1016.47
	5	7.53	157.55	1464.60	2330.01
	10	8.20	277.37	2289.20	3059.75
	15	8.13	336.80	3489.20	3716.28
30	1	7.27	51.30	553.74	1359.42
	5	7.93	170.41	1883.7	2226.07
	10	8.27	364.15	3198.40	2678.38
	15	8.27	430.42	4776.6	4852.21

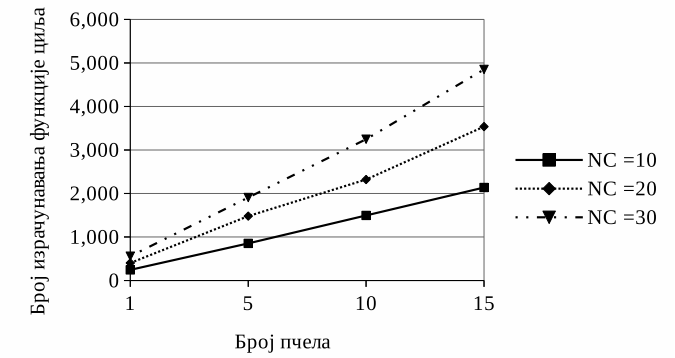


Слика 5.1: Просечан број успешних стартовања у зависности од параметара B и NC

Просечан број успешних стартовања, очигледно, расте са повећањем вредности параметара B и NC, док не достигне одређени ниво. Просечно време успешних итерација и просечан број израчунавања функције циља, такође, расту са порастом вредности параметара B и NC, што је и очекивано. Највећи просечан број успешних стартовања, међу свим (B, NC) комбинацијама, имају комбинације (10,20), (10,30),



Слика 5.2: Просечно време извршавања успешне итерације у зависности од параметара B и NC

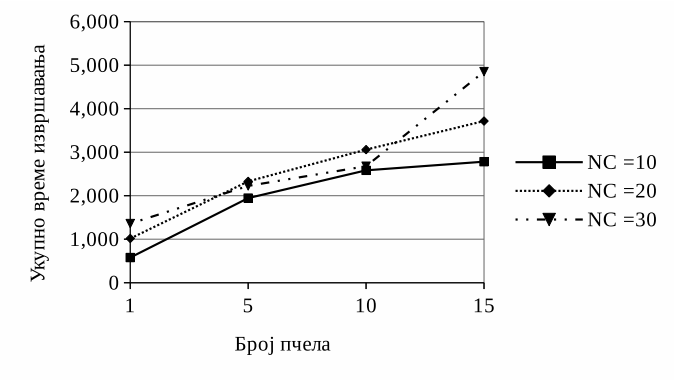


Слика 5.3: Просечан број израчунавања функције циља у зависности од параметара B и NC

(15,20), (15,30), а међу њима најмање просечно време извршавања има комбинација (10,30). Из наведених разлога, за тестирање свих генерисаних формула изабране су вредности $NC = 30$ и $B = 10$.

Резултати добијени тестирањем свих генерисаних формула приказани су у табели 5.3. Колона Решено садржи успешност у доказивању тестиране формуле у 10 независних стартовања под истим условима. Трећа колона садржи просечан број израчунавања вредности функције циља у свих 10 поновних стартовања. Последња колона садржи просечно време извршавања сваког стартовања, укључујући и неуспешна стартовања.

Као што се може видети из резултата приказаних у табели 5.3, BCOi метода показује веома стабилне перформансе. Од 39 тестираних формула за 8 није успела да докаже задовољивост. У већини осталих случајева сва стартовања су била успешна, тј. у 10 од 10 стартовања доказана је задовољивост формула. Код примера где нису сва стратовања била успешна, време извршавања се битно повећава, због неуспешних покушаја у којима је извршено свих 50 итерација. Такође, време извршавања није превише велико у случајевима са 100% успешности, што иде у прилог закључку у



Слика 5.4: Просек укупног времена извршавања сваког стартовања у зависности од параметара B и NC

Табела 5.3: BCOi резултати за све генерисане формуле

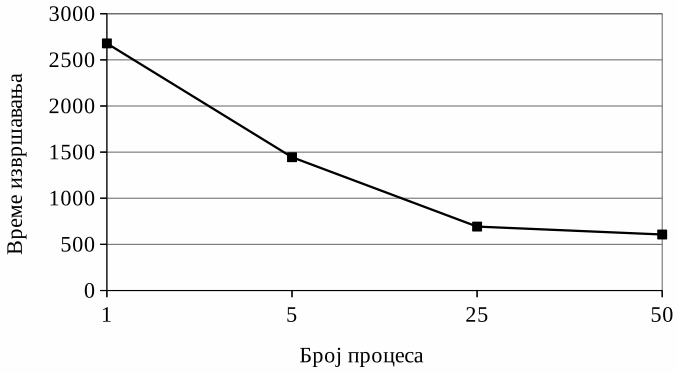
N, L, Пример	Решено	Бр. израч. ф-ја циља	CPU време
10, 30, 0	10/10	10	7.768
10, 30, 1	10/10	10	8.000
10, 30, 2	4/10	11946	1819.987
10, 50, 0	10/10	34	82.972
10, 50, 1	10/10	178	378.609
10, 50, 2	10/10	19	41.336
10, 100, 0	0/10	15000	–
10, 100, 1	10/10	20	208.448
10, 100, 2	10/10	10	14.491
20, 20, 0	10/10	17	3.856
20, 20, 1	10/10	51	16.585
20, 20, 2	10/10	138	59.054
20, 50, 0	10/10	32	44.816
20, 50, 1	10/10	10	19.229
20, 50, 2	9/10	6651	10771.538
20, 100, 0	10/10	120	128.694
20, 100, 1	10/10	56	541.613
20, 100, 2	10/10	10	75.885
20, 250, 0	10/10	360	7571.212
20, 250, 1	0/10	15000	–
20, 250, 2	0/10	15000	–
30, 30, 0	0/10	15000	–
30, 30, 1	10/10	40	18.167
30, 30, 2	10/10	83	53.004
30, 60, 0	10/10	59	153.523
30, 60, 1	10/10	150	408.450
30, 60, 2	10/10	24	74.316
30, 100, 0	0/10	15000	–
30, 100, 1	10/10	1260	4497.525
30, 100, 2	10/10	480	3793.997
30, 250, 0	10/10	10	442.865
30, 250, 1	10/10	62	3369.639
30, 250, 2	0/10	15000	–
40, 40, 0	0/10	15000	–
40, 40, 1	10/10	14	8.922
40, 40, 2	10/10	360	412.283
50, 50, 0	0/10	15000	–
50, 50, 1	10/10	3420	4118.458
50, 50, 2	3/10	11517	23622.824

раду [76] да је механизам модификације решења кључан за ефикасност BCOi методе.

5.3 Паралелизација BCOi алгоритма

Општи начин рада BCOi алгоритма је врло повољан за конструкцију дистрибуираног доказивача. За најприхватљивији метод, због своје флексибилности, је процењена MPI метода паралелизације[1]. Обзиром да се читав алгоритам одвија по итерацијама које се извршавају независно, најприродније је било паралелним процесима доделити одређен број итерација које треба извршити. Да је паралелизација вршена по вештачким пчелама, пчеле-процеси би морале да након сваког лета унапред размене податке о вредности функције циља, донесу одлуку о верности досадашњем решењу и уколико је потребно преузму решење од пчеле регрутера. Овакав начин организације би битно успоравао рад, јер би у том случају било неопходно синхронизовати процесе и за сваку размену порука било би неопходно чекати да све пчеле заврше лет унапред. Такође, број паралелних процеса би био ограничен бројем пчела, а секвенцијални алгоритам је показао да велико повећање броја пчела не доприноси ефикасности.

Паралелизацијом по итерацијама није било потребно остваривати било какву комуникацију међу процесима, ни размењивати податке током рада, што знатно убрзава извршавање. Додатно, када неки од процеса током извршавања пронађе решење, он шаље сигнал и сви остали процеси прекидају рад. За потребе тестирања изабран је исти скуп примера на којима је тестирана и најефикаснија стратегија секвенцијалног приступа, а за параметре су изабране исте вредности $NC = 30$ и $B = 10$. У овом случају је, такође, број итерација ограничен на 50. Добијени резултати су приказани на слици 5.5.



Слика 5.5: Просечно време извршавања у зависности од броја паралелних процеса

Тестирања су вршена на кластеру који садржи 22 сервера, сваки са по два Intel

[1]Кратак опис MPI паралелизације дат је у поглављу А.3

Xeon E5 - 2670 процесора са 8 језгара и 64GB меморије (4GB RAM по језгру) под оперативним системом Scientific Linux 6.5 64-bit са верзијом компајлера gcc ver. 4.4.

Приликом тестирања је мерено само просечно време извршавања за различит број паралелних процеса у 10 независних покретања сваког примера. Укупан број итерација није промењен у односу на секвенцијално извршавање, па није очекивано ни да се промени успешност. Обзиром на број језгара по процесору и укупан број итерација које треба поделити равномерно, за број паралелних процеса је узето 1, 5, 25 и 50. Као што је и очекивано, са повећањем броја паралелних процеса добија се велико убрзање. Линеарна зависност између броја процесора и времена извршавања изостаје, што је и очекивано, пошто се и приликом секвенцијалног извршавања не изврши сваки свих 50 итерација, већ се извршавање прекида када се пронађе решење. Повећање са 25 на 50 паралелних процеса не доноси велико побољшање, што је у складу и са проценом да је приликом секвенцијалног приступа је ограничење на 50 итерација сасвим довољно. Обзиром на добијене резултате, може се закључити да се паралелизацијом на 25 процеса добијају најбољи резултати.

Глава 6

BCOi приступ у дифолтном резоновању

У делу 3.3 је детаљно описан метод за добијање система неједнакости за дати проблем дифолтног резоновања. За проналажење решења добијеног система коришћене су различите стратегије и упоређени добијени резултати.

6.1 Реализација BCOi алгоритма за примену у дифолтном резоновању

Као и у претходном поглављу узима се да вектор решења, ако постоји, има највише $L+1$ вредност већу од нуле, где је L број неједнакости у добијеном систему ([23, 28, 50]). Решење је, дакле, низ који садржи $L+1$ атом

$$a = [a_1, a_2, \ldots, a_{L+1}]$$

где су a_i, $i = 1, \ldots, L+1$ атоми из скупа свих атома At, којима су додељене вероватноће

$$x = [x_1, x_2, \ldots, x_{L+1}]\,.$$

Вероватноће атома који не припадају низу a имају вредност 0. Ако је познато решење $a = [a_1, a_2, \ldots, a_{L+1}]$ тада се систем може записати у облику

$$\sum_{j=1}^{L+1} c_{ij} x_i \ \rho_i \ 0, \qquad i = 1, \ldots, L \tag{6.1.1}$$

где су c_{ij} $(i = 1, \ldots, L, j = 1, \ldots, L+1)$ коефицијенти и $\rho_i \in \{\leqslant, <, \geqslant, >\}$ $(i = 1, \ldots, L)$.

Имплементирани BCOi метод се састоји из четири фазе. Прва фаза представља генерисање иницијалног решења на почетку сваке итерације. У другој фази се врши модификација решења. Трећа фаза је за поређење решења, а четврта за регрутацију.

Трећа и четврта фаза се примењују на потпуно исти начин, као и у случају примене BCOi методе на испитивање задовољивости вероватносних формула, па овде неће бити поново објашњаване.

6.1.1 Иницијално решење

Свака итерација BCOi алгоритма почиње генерисањем иницијалног решења за сваку пчелу. Свака пчела случајно бира $M = \min\{2^N, L+1\}$ атома којима ће доделити не-нула вредности вероватноћа. Највише $E = M/2$ случајно одабраних атома ће имати вероватноћу ε, а преосталих $M-E$ атома ће имати вероватноћу $\frac{1-E\cdot\varepsilon}{M-E}$. Овакво генерисање иницијалног решења се дешава на почетку сваке итерације у случају да се из претходне итерације не преносе нека решења. Обзиром да је тестирана и стратегија преношења дела решења из једне итерацију у другу, механизам одабира решења ће бити накнадно објашњен. Пчеле које нису преузеле решење из претходне итерације и у овом случају, своје решење генеришу на описан начин.

6.1.2 Модификација решења

Модификација решења током сваког од NC летова унапред је главни корак у BCOi алгоритму. Функција циља се, исто као и у претходном поглављу, дефинише као сума растојања леве стране сваке неједнакости система (6.1.1) од нуле. Нека је x тренутно решење, функција циља z се дефинише као

$$z(x) = \sum_{i=1}^{L} d_i(x), \tag{6.1.2}$$

где је d_i растојање леве стране неједнакости i од нуле, дефинисано помоћу релације

$$d_i(x) = \begin{cases} (\sum_{j=1}^{M} c_{ij}x_j)^2 & \text{ако неједнакост } i \text{ није задовољена,} \\ 0 & \text{у супротном,} \end{cases} \tag{6.1.3}$$

На почетку сваког лета унапред, уколико је број атома M у тренутном решењу мањи од укупног броја атома 2^N, у скуп атома решења се додаје нових $L/5$ атома са додељеним вероватноћама $1/(L+1)$. У решењу се задржава M атома са најбољом оценом. Оцена атома, као и у претходном поглављу, представља меру значајности атома у систему (6.1.1) и одређује се на исти начин. Дакле, атому a_j ($j \in \{1, \ldots M\}$) одговара колона $c_j = [c_{1j}, c_{2j}, \ldots, c_{Lj}]^T$ система (6.1.1) и његова оцена се одређује по формули

$$grade(a_j) = \sum_{i=1}^{L} c_{ij} \cdot \mathrm{sgn}(i),$$

где је $\mathrm{sgn}(i)$ знак i-те неједнакости систма, тј.

$$\mathrm{sgn}(i) = \begin{cases} 1, & \text{if } \rho \in \{\geqslant, >\}, \\ -1, & \text{if } \rho \in \{\leqslant, <\}. \end{cases}$$

Вредност функције циља је ненегативна, а циљ је да постане 0, уз ограничења да свака од вероватноћа x_i мора бити ненегативна и да је збир свих вероватноћа 1. Међутим, није довољно пронаћи решење за које ће вредност функције циља бити 0, већ и свака од неједнакости у систему мора бити задовољена, обзиром да се у систему налазе и неједнакости у којима се као симболи релације налазе $<$ и $>$. Уколико лева страна ових неједнакости има вредност 0, неједнакост није задовољена.

Уколико решење није пронађено, као и у претходном поглављу, биће коришћене хеуристике WUP и GG, које су раније објашњене, уз додатак Нелдер-Мид методе нелинеарног програмирања (енг. *Nelder-Mead nonlinear programming method*), која је детаљно објашњена у поглављу A.2. За проналажење најбоље стратегије за решавање система биће прављене различите комбинације наведених метода. Без обзира на комбинацију метода, поправљање решења биће вршено одмах након генерисања иницијалног решења, а затим у сваком лету унапред. Нелдер-Мид метода је коришћена самостално или у комбинацији са једном или обе наведене хеуристике, уколико функција циља (6.1.2) није поправљена у P узастопних летова унапред ($P \in \{1, 5, 10\}$). Нелдер-Мид метода је сваки пут покретана 10 пута.

Након сваке модификације вероватноћа атома неопходно је нормализовати добијене вредности пре него што се настави са алгоритмом.

6.1.3 Стратегије коришћене за проналажење најефикасније методе решавања проблема

Итерација се зауставља уколико се пронађе решење (све неједнакости ситема су задовољене) или уколико је извршено NC летова унапред и уназад. Уколико решење није пронађено коришћене су две стратегије за генерисање иницијалног решења у наредној итерацији. Прва стратегија је да се сва тренутна решења обришу и да наредна итерација почне генерисањем потпуно нових решења, на претходно објашњен начин. У другој стратегији се све време чува најбоље пронађено решење, тј. решење са најмањом вредношћу функције циља. У сваком лету унапред се проверава да ли је нека од пчела пронашла боље решење које постаје тренутно најбоље решење. Када се итерација заврши, од свих тренутних решења, укључујући и најбоље, случајно се бира највише $B/3$ решења и та решења се користе као иницијална решења у наредној итерацији. Иницијална решења за преостале пчеле у наредној итерацији се генеришу на описани начин.

Са друге стране, приликом модификације решења, коришћена су два различита приступа. У првом приступу, уколико се добије лошије решење у односу на тренутно,

тј. дође до повећања вредности функције циља, задржава се старо решење. У другом приступу се не проверава на који начин се променила вредност функције циља, већ се у наставку увек користи решење добијено модификацијом.

BCOi итерације се заустављају када се испуни критеријум за заустављање, што је у овом случају пронађено решење система или достигнут максималан предвиђен број итерација. Резултати добијени тестирањем су дати у наставку.

6.2 Анализа и поређење резултата добијених тестирањем

За потребе тестирања, из литературе ([5, 9, 59]) је издвојено 18 примера дифолтног закључивања, на основу чега је добијено 60 различитих формула $LPCP$ логике чија се задовољивост тестира. У изабраним примерима, постоје два типа: примери код којих $\Delta \mid\!\sim \alpha \rightarrowtail \beta$ и они код којих $\Delta \not\mid\!\sim \alpha \rightarrowtail \beta$ и $\Delta \not\mid\!\sim \alpha \rightarrowtail \neg\beta$.

Све вероватноће и константе у добијеном систему неједнакости су из Хардијевог поља $\mathbb{Q}(\varepsilon, K)$. Ове вредности су представљене као рационалне функције које зависе од ε и K са реалним коефицијентима двоструке тачности, тј.

$$\eta = \frac{\sum_{j=0}^{e} \left(\sum_{k=0}^{l_j} h_{jk} K^k \right) \varepsilon^j}{1 + \sum_{j=1}^{e} \left(\sum_{k=0}^{l_j} h_{jk} K^k \right) \varepsilon^j}$$

У оваквом представљању максимум параметра e је 4 (користећи исту идеју као и у раду [93]). Првим тестирањима се показало да примена WUP хеуристике за модификацију решења у сваком кораку повећава максимални степен l_j полинома $\sum_{k=0}^{l_j} h_{jk} K^k$ који се налази уз ε^j. Ови полиноми већ након прве модификације решења постају јако велики и заузимају доста меморије и свака манипулација њима постаје немогућа. У поглављу 3.3 већ је речено да је $K^k \cdot \varepsilon \approx 0$, па користећи то ови полиноми, за $j > 0$, су апроксимирани вредношћу $h_{jl_j} K^0$. На овај начин је избегнут експоненцијални раст ових полинома, при чему и полином уз ε^0 задржава мали максимални степен l_0. Повећање степена l_j уз $\varepsilon^j, j \geqslant 0$ се не дешава приликом примене хеуристике GG и Нелдер-Мид методе, па није било потребе за било каквим модификацијама добијених вредности.

Након сваке промене вероватноћа атома, неопходно је нормализовати добијене вредности. Приликом нормализације, вредности које су блиске 0 или 1, доводе до повећања коефицијената h_{jk}, при чему ове вредности након неколико итерација излазе из опсега реалних бројева двоструке прецизности. Из овог разлога, приликом нормализације, вредности ових коефицијената су ограничене на $\pm 10^{10}$, осим вредности коефицијента h_{0k}.

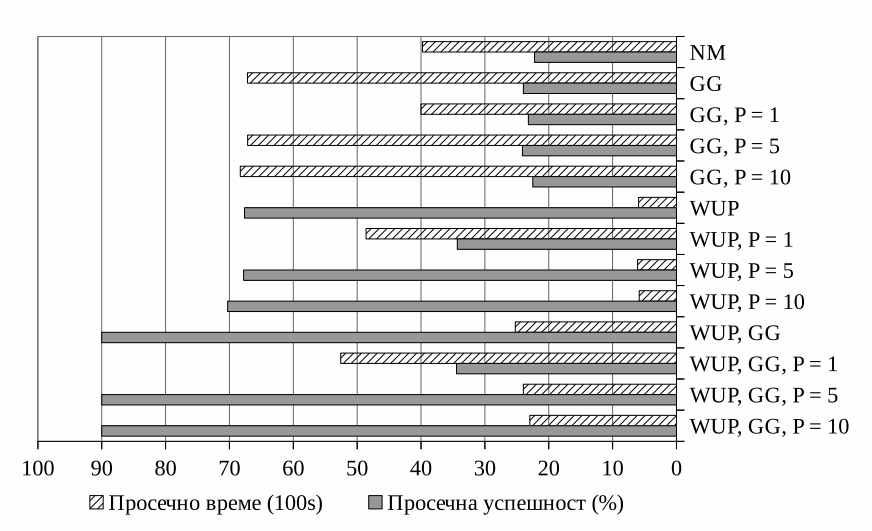
Све имплеметације су тестиране на кластеру који садржи 22 сервера, сваки са по два Intel 2.6GHz процесора са 8 језгара и 64GB меморије (4GB RAM по језгру) под оперативним системом Scientific Linux 6.5 64-bit са верзијом компајлера gcc ver. 4.4.

Као што је већ речено, за модификацију решења коришћене су различите комбинације поменутих метода. Коришћене су самостално WUP хеуристика, GG хеуристика, Нелдер-Мид метода (NM), и комбинације две или све три методе. У ситуацијама када је Нелдер-Мид метода комбинована са неком хеуристиком, ова метода није примењивана у сваком лету напред, већ уколико вредност функције циља (6.1.2) није поправљена у P узастопних корака, при чему је варирана вредност параметра P ($P = \{1, 5, 10\}$). За поређење ових стратегија из скупа свих формула случајно је изабрано 24 задовољиве формуле и мерен просечан проценат успешности у 30 независних извршавања и просечно време трајања ових извршавања (успешних и неуспешних). Добијени резултати су приказани у табели 6.1 и на слици 6.1 (на слици 6.1 јединично време је 100 секунди). На основу резулата примене BCOi методе на формуле $LPCP$ логике [111], за потребе овог поређења узете су следеће вредности параметара $NC = 30$, $B = 10$, максималан број итерација 50, у свакој итерацији се генерише ново иницијално решење, при свакој модификацији решења задржава се решење са мањом вредношћу функције циља. За хеуристике WUP и GG су коришћени параметри дати у поглављу 5.1.

Табела 6.1: Поређење различитих метода модификације решења

	Просечан проценат успешности (%)	Просечно време извршавања (s)
NM	22.22	3978.00
GG	23.97	6718.00
GG, P = 1	23.19	4001.00
GG, P = 5	24.12	6715.00
GG, P = 10	22.50	6830.00
WUP	67.64	592.00
WUP, P = 1	34.31	4861.00
WUP, P = 5	67.78	609.00
WUP, P = 10	70.28	583.00
WUP, GG	90.00	2524.00
WUP, GG, P = 1	34.44	5258.00
WUP, GG, P = 5	90.00	2397.00
WUP, GG, P = 10	90.00	2296.00

Као што се види из табеле 6.1 хеуристика WUP даје највећи допринос у проналажењу решења. Најбољи резултати се добијају када се комбинују све три методе, међутим, уколико се Нелдер-Мид метода користи превише често ($P = 1$), резултати изостају. Ако се као мера ефикасности користи однос времена извршавања и проценат успешности, тада је најефикасније користити комбинацију све три методе са параметром $P = 10$. Међутим, за потребе даљег тестирања коришћена је стратегија која је дала најмање време извршавања, тј. хеуристика WUP комбинована са Нелдер-Мид методом за $P = 10$, у очекивању да ће бољи избор параметара B и



Слика 6.1: Поређење различитих метода модификације решења

NC, другачији приступ у генерисању иницијалног решења и модификацији решења повећати ефикасност.

За избор најбоље стратегије решавања проблема, максималан број итерација је ограничен на 50. Мерен је однос времена извршавања и процента успешности у случају када се у свакој итерацији генеришу нова иницијална решења, насупрот варијанти када се највише $B/3$ случајно изабраних решења (од B постојећих решења са додатком тренутно најбољег решења) преноси као иницијално решење за наредну итерацију. Разматране су варијанте када се приликом модификације решења увек задржава боље решење, насупрот варијанти у којој је дозвољено кварење функције циља. Ово је дало четири стратегије које су комбиноване са варирањем вредности параметара B и NC. Добијени резултати су приказани у табелама 6.2 и 6.3 и на сликама 6.2 и 6.3.

На основу датих резултата, види се да уколико се не дозволи повећање вредности фукције циља приликом модификације решења, повећање параметра B доводи до повећања времена извршавања и благог повећавања успешности, док повећање параметра NC доводи до повећања успешности и том проликом долази до смањења или благог повећања времена извршавања. Преношење дела решења из једне итерације у другу, у овом случају, доводи до великог повећања ефикасности. Са друге стране, ако се дозволи повећање вредности фукције циља, добија се потпуно другачија ситуација. У овом случају просечна успешност за већину комбинација параметара B и NC је 100% или близу томе и из тог разлога је одвојено посматрано само просечно време извршавања. Уколико свака итерација почиње са новим иницијалним решењима, са повећањем параметра NC до одређене вредности, смањује се време извршавања, након чега постаје готово константно. У случају да се део решења пренесе из претходне итерације, са повећањем параметра NC повећава се време извршавања, осим у ситуацији $B = 1$ када преношење решења не постоји. На основу свега реченог, нај-

Табела 6.2: Резултати добијени варирањем параметара B и NC у случају када није дозвољено повећање вредности функције циља

B	NC	Генерисање нових иницијални решења		Преношење дела решења из претх. итер.	
		Просечна успешност (%)	Просечно време (s)	Просечна успешност (%)	Просечно време (s)
1	10	26.39	25.08	26.39	25.08
	20	41.94	41.10	41.94	41.10
	30	50.14	52.31	50.14	52.31
	40	54.31	64.34	54.31	64.34
	50	62.08	72.94	62.08	72.94
	60	66.11	82.52	66.11	82.52
	70	72.78	85.17	72.78	85.17
5	10	31.25	121.70	62.64	69.56
	20	49.31	187.27	89.44	43.48
	30	60.28	234.77	94.86	41.48
	40	65.83	291.63	99.03	18.60
	50	74.03	306.42	98.75	42.61
	60	80.56	333.61	99.72	30.58
	70	89.03	314.68	99.86	36.95
10	10	35.00	230.35	74.44	94.01
	20	55.07	335.70	98.61	22.45
	30	63.33	445.47	99.17	26.39
	40	74.53	485.30	98.61	51.22
	50	81.53	504.76	99.86	33.21
	60	89.17	484.83	100.00	43.08
	70	94.20	445.85	99.86	55.31
15	10	35.83	335.88	86.39	83.83
	20	52.53	528.45	99.31	26.69
	30	68.33	631.93	99.58	34.02
	40	77.08	685.42	99.44	50.97
	50	88.19	631.23	99.86	45.69
	60	90.93	677.28	100.00	45.78
	70	97.36	526.23	100.00	54.10
20	10	37.50	432.92	82.22	134.46
	20	54.72	659.20	95.28	90.23
	30	71.11	748.89	100.00	27.81
	40	79.58	833.66	99.86	38.49
	50	90.42	701.83	100.00	44.33
	60	95.69	632.13	100.00	45.14
	70	97.92	547.45	100.00	55.93

ефикаснија стратегија је да се дозволи повећање вредности фукције циља приликом модификације решења и да се део решења преноси из претходне итерације, при чему се узимају вредности параметара $B = 5$ и $NC = 20$. Ова комбинација стратегија и параметара је коришћена за тестирање преосталих формула. Разлог да се не користи вредност параметара $NC = 10$ је тај што је просечна успешност у том случају нешто мања од 100%. На основу овога, најбоља стратегија за решавање посматраног проблема је дата псеудокодом 6.1.

За сваки пример је, такође, тестирана задовољивост добијене формуле $LPCP$ логике, тј. задовољивост добијеног система неједнакости, применом FME методе. У табели 6.4 су приказани резултати тестирања примера код којих $\Delta \mid\!\sim \alpha \rightarrowtail \beta$, док су у табелама 6.5 и 6.6 приказани резултати тестирања за примере код којих $\Delta \not\mid\!\sim \alpha \rightarrowtail \beta$ и $\Delta \not\mid\!\sim \alpha \rightarrowtail \neg\beta$.

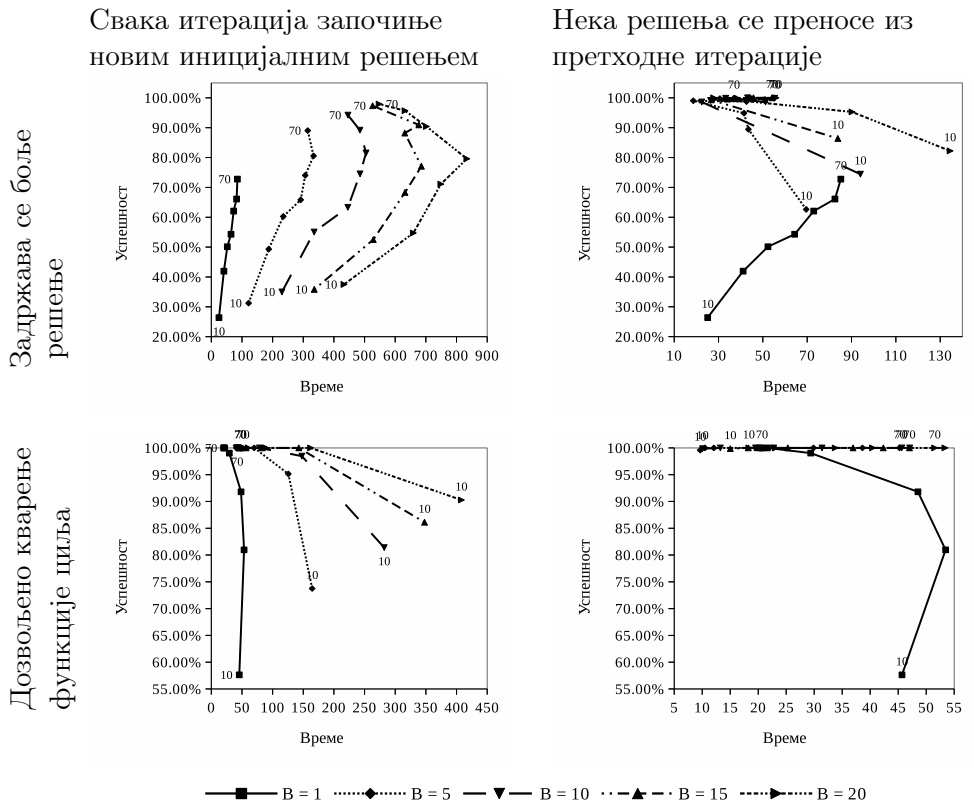
За сваки пример је тестирана задовољивост три формуле (Δ, $\Phi_1 = \Delta \cup \{\alpha \rightarrowtail \beta\}$ и $\Phi_2 = \Delta \cup \{\neg(\alpha \rightarrowtail \beta)\}$) коришћењем BCOi методе (врсте BCOi) и FME методе (врсте

Табела 6.3: Резултати добијени варирањем параметара B и NC у случају када је дозвољено повећање вредности функције циља

B	NC	Генерисање нових иницијални решења		Преношење дела решења из претх. итер.	
		Просечна успешност (%)	Просечно време (s)	Просечна успешност (%)	Просечно време (s)
1	10	57.64	45.68	57.64	45.68
	20	80.97	53.47	80.97	53.47
	30	91.81	48.53	91.81	48.53
	40	99.03	29.38	99.03	29.38
	50	100.00	21.65	100.00	21.65
	60	100.00	20.15	100.00	20.15
	70	100.00	20.67	100.00	20.67
5	10	73.75	164.55	99.58	9.64
	20	95.14	125.73	100.00	10.49
	30	100.00	69.96	100.00	12.07
	40	100.00	48.44	100.00	19.83
	50	100.00	42.19	100.00	29.88
	60	100.00	40.99	100.00	38.62
	70	100.00	47.97	100.00	45.24
10	10	81.39	282.05	99.86	10.06
	20	98.47	147.70	100.00	13.23
	30	100.00	78.59	100.00	19.53
	40	100.00	42.87	100.00	22.76
	50	100.00	40.50	100.00	31.39
	60	100.00	41.85	100.00	47.06
	70	100.00	42.42	100.00	45.62
15	10	86.11	347.80	99.86	15.02
	20	100.00	142.71	100.00	18.24
	30	100.00	83.20	100.00	20.70
	40	100.00	56.84	100.00	25.29
	50	100.00	48.19	100.00	36.94
	60	100.00	45.79	100.00	42.34
	70	100.00	47.05	100.00	47.01
20	10	90.28	407.93	100.00	18.33
	20	100.00	162.07	100.00	19.79
	30	100.00	89.53	100.00	23.05
	40	100.00	60.40	100.00	33.76
	50	100.00	51.61	100.00	40.43
	60	100.00	53.88	100.00	53.34
	70	100.00	52.56	100.00	51.51

FME). Прва колона за сваку формулу за BCOi представља успешност у 30 независних извршавања, друга колона је просечно време извршавања (у секундама). Прва колона за сваку формулу за FME методу садржи информацију да ли је задовољивост доказана или не ("sat" или "un-sat") или нема довољно меморије ("MEM"), у другој колони се налази време извршавања (у секундама). Као што се може видети, FME метода у већини случајева не може да заврши израчунавање, док BCOi има велики проценат успешности (осим код формула Φ_2 у табели 6.4 које нису задовољиве) са знатно мањим временом извршавања.

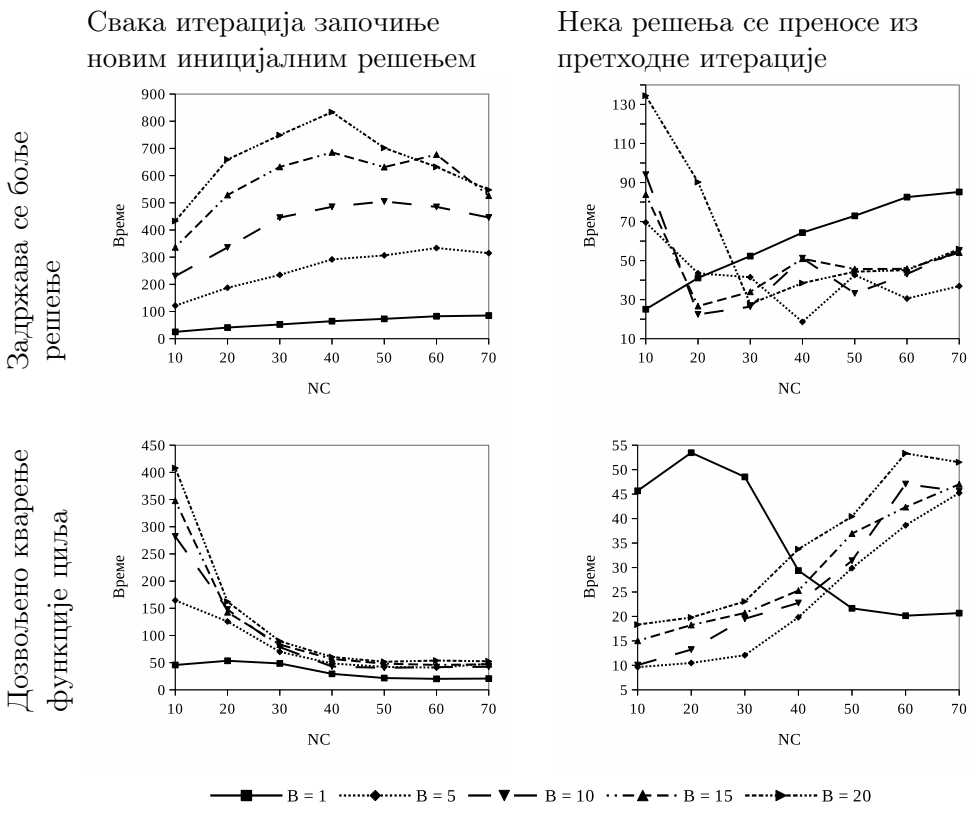
Разлог за неуспех FME методе је експоненцијални раст броја неједнакости током елиминације непознатих из система. На пример, за тестирање задовољивости скупа формула $\{bird \rightarrowtail fly, penguin \rightarrowtail bird, penguin \rightarrowtail \neg fly, fly \rightarrowtail \neg penguin\}$, формира се систем који садржи 30 неједнакости. Након елиминације прве непознате, у систему остаје 26 неједнакости, након елиминације друге непознате систем има 29 неједнакости, а затим 38, 63, 229, 5494 неједнакости. Након елиминације седме непознате



Слика 6.2: Однос успешности и времена извршавања у зависности од параметара B и NC

систем има око $2 \cdot 10^6$ неједнакости. Још драстичнија ситуација се добија приликом тестирања задовољивости скупа формула $\{\theta \rightarrowtail \phi, \theta \rightarrowtail \psi\}$ када у полазном систему има 24 неједнакости, а затим након елиминације сваке следеће непознате систем има 21, 30, 39, 118, 444, 37014, да би након елиминације седме непознате систем достигао величину од $6 \cdot 10^6$ неједнакости. У оваквим ситуацијама се извршавање алгоритма зауставља због недовољне RAM меморије.

Формуле Φ_2 из табеле 6.4 су најинтересантније. Познато је да ове формуле нису задовољиве, што подаци добијени применом BCOi методе потврђују. Међутим, уколико се овај метод користи за испитивање задовољивости неке случајне формуле, податак да BCOi метод не може да нађе решење система даје велику шансу тврђењу да формула није доказива, међутим и даље постоји могућност да BCOi метода не може да нађе решење иако је формула задовољива. Слична ситуација се добија и ако се FME метода примени на ову формулу, при чему се неки атоми проглашавају небитним, као у примеру 3.3.1 у поглављу 3.3. Ово значи да ће полазни систем неједнакости бити мањи, обзиром да неке непознате добијају вредност 0 и тиме се елиминишу из система. Међутим, обзиром да се не испитује задовољивост читавог система, добијени податак да је систем незадовољив је само смерница у доношењу закључака. Најбољи резултати добијени применом FME методе на овај начин при-



Слика 6.3: Време извршавања у зависности од параметра NC за различите вредности параметра B

казани су у табели 6.7.

Као и у претходним табелама, у табели 6.7 у првој и четвртој колони дати су тестирани примери, у другој и петој колони добијени резултати, а у трећој и шестој колони најбоља времена извршавања. Обзиром да су у овом случају за сваки пример тестиране све могуће комбинације атома који се проглашавају небитним, за сваки пример добијено је по више резултата. У табели су приказана најбоља добијена времена, а чак и у оваквом приступу има примера код којих FME алгоритам није могао да заврши израчунавање због недостатка меморије.

На основу добијених укупних резултата, може се закључити да се BCOi метода може користити као поуздан алат у испитивању задовољивости скупа дифолта. Чињенице које иду томе у прилог су велики проценат успешности у доказивању задољивости скупа дифолта и релативно кртако време извршавања. Са друге стране, у извођењу закључка да је неки дифолт последица базе дифолта, информација да BCOi метода не може да докаже задовољивост формуле Φ_2, заједно да информацијом да се применом FME методе на редукован систем, добијен на основу формуле Φ_2, показује незадовољивост система, се може искористити као поуздана смерница.

Алгоритам 6.1: BCOi алгоритам за задовољивост у дифолтној логици

BCOi(B, NC)
  $Iter = 1$; $found =$ true;
  $keep = 0$; $NM = 0$;
  Do
   // Генерисање колоније
   For $(b = 0; b < B; b++)$
    If $(b > keep - 1)$ Случајно генерисање M атома и додела вероватноћа;
    Поправљање решења коришћењем WUP хеуристике;
    If (решење пронађено) **return** $found$;
   For $(s = 0; s < NC; s++)$
   //Лет унапред
    For $(b = 0; b < B; b++)$
     Додавање нових $L/5$ атома са вероватноћама $1/(L+1)$;
     Оцена свих атома и избор $L+1$ најбољих;
     Поправљање решења коришћењем WUP хеуристике;
     If $(NM = P)$ примена Нелдер-Мид методе;
     If (решење поправљено) $NM = 0$ **else** $NM++$;
     If (решење пронађено) **return** $found$;
   //Лет уназад
    For $(b = 0; b < B; b++)$
     Оцена решења за пчелу b коришћењем (5.1.4);
    For $(b = 0; b < B; b++)$
     Одлука о лојалности решењу за пчелу b коришћењем (5.1.5);
    For $(b = 0; b < B; b++)$
     If (b је неопредељена) случајан избор регрутера коришћењем (5.1.6);
    Избор најбољег решења;
    Случајно генерисање вредности $keep \in [0, B/3]$;
    Случајан избор $keep$ решења;
   $Iter++$;
  While $(Iter \leqslant Iter_{max})$;
  return not$(found)$;

Табела 6.4: Резултати тестирања примера код којих $\Delta \mid\!\sim \alpha \rightarrowtail \beta$

Пример	Метода	Δ		Φ_1		Φ_2	
		Статус	Време	Статус	Време	Статус	Време
$\dfrac{bird \rightarrowtail fly \quad penguin \rightarrowtail bird \quad penguin \rightarrowtail \neg fly}{bird \wedge penguin \rightarrowtail \neg fly}$	BCOi	30/30	6.34	30/30	4.44	0/30	–
	FME	sat	79957.29	sat	124076.15	un-sat	626.75
$\dfrac{bird \rightarrowtail fly \quad penguin \rightarrowtail bird \quad penguin \rightarrowtail \neg fly}{fly \rightarrowtail \neg penguin}$	BCOi	30/30	6.34	30/30	16.89	0/30	–
	FME	sat	79957.29	MEM	–	MEM	–
$\dfrac{bird \rightarrowtail fly \quad penguin \rightarrowtail bird \quad penguin \rightarrowtail \neg fly}{bird \rightarrowtail \neg penguin}$	BCOi	30/30	6.34	30/30	6.08	0/30	–
	FME	sat	79957.29	MEM	–	MEM	–
$\dfrac{bird \rightarrowtail fly \quad penguin \rightarrowtail bird \quad penguin \rightarrowtail \neg fly}{bird \vee penguin \rightarrowtail \neg penguin}$	BCOi	30/30	6.34	30/30	12.89	0/30	–
	FME	sat	79957.29	MEM	–	MEM	–
$\dfrac{bird \rightarrowtail fly \quad penguin \rightarrowtail bird \quad penguin \rightarrowtail \neg fly}{bird \vee penguin \rightarrowtail fly}$	BCOi	30/30	6.34	30/30	10.74	0/30	–
	FME	sat	79957.29	MEM	–	MEM	–
$\dfrac{\theta \rightarrowtail \phi \quad \theta \rightarrowtail \psi}{\theta \rightarrowtail \phi \wedge \psi}$	BCOi	30/30	0.80	30/30	0.46	0/30	–
	FME	MEM	–	MEM	–	MEM	–
$\dfrac{\theta \rightarrowtail \phi \quad \phi \rightarrowtail \psi}{\theta \vee \phi \rightarrowtail \psi}$	BCOi	30/30	0.48	30/30	0.32	0/30	–
	FME	MEM	–	MEM	–	MEM	–
$\dfrac{\theta \rightarrowtail \psi \quad \theta \rightarrowtail \phi}{\theta \wedge \phi \rightarrowtail \psi}$	BCOi	30/30	0.80	30/30	0.26	0/30	–
	FME	MEM	–	MEM	–	MEM	–
$\dfrac{\theta \rightarrowtail \phi \quad \theta \wedge \phi \rightarrowtail \psi}{\theta \rightarrowtail \psi}$	BCOi	30/30	0.18	30/30	0.61	0/30	–
	FME	sat	115.77	MEM	–	MEM	–
$\dfrac{\theta \rightarrowtail \phi \quad \theta \wedge \phi \rightarrowtail \psi}{\theta \rightarrowtail \psi}$	BCOi	30/30	0.18	30/30	0.83	0/30	–
	FME	sat	115.77	MEM	–	MEM	–
$\dfrac{\theta \rightarrowtail \phi \quad \phi \rightarrowtail \theta \quad \theta \rightarrowtail \psi}{\phi \rightarrowtail \psi}$	BCOi	30/30	1.85	30/30	1.95	0/30	–
	FME	MEM	–	MEM	–	MEM	–
$\dfrac{\theta \rightarrowtail \phi \rightarrow \psi \quad \theta \rightarrowtail \phi}{\theta \rightarrowtail \psi}$	BCOi	30/30	0.22	30/30	0.76	0/30	–
	FME	MEM	–	MEM	–	MEM	–

Табела 6.5: Резултати тестирања за примере код којих $\Delta \not\sim \alpha \rightarrowtail \beta$ и $\Delta \not\sim \alpha \rightarrowtail \neg\beta$

Пример	Метода	Δ		Φ_1		Φ_2	
		Статус	Време	Статус	Време	Статус	Време
$bird \rightarrowtail fly$ $penguin \rightarrowtail bird$ $penguin \rightarrowtail \neg fly$ $metalWings \rightarrowtail fly$ $yogi \rightarrowtail fly$ $bird \wedge penguin \wedge metalWings \rightarrowtail \neg yogi \vee fly$	BCOi	30/30	29.52	28/30	148.14	30/30	26.91
	FME	MEM	–	MEM	–	MEM	–
$bird \rightarrowtail fly$ $penguin \rightarrowtail bird$ $penguin \rightarrowtail \neg fly$ $metalWings \rightarrowtail fly$ $yogi \rightarrowtail fly$ $bird \wedge penguin \wedge metalWings \rightarrowtail yogi \wedge \neg fly$	BCOi	30/30	29.52	29/30	91.01	30/30	43.58
	FME	MEM	–	MEM	–	MEM	–
$penguin \rightarrowtail featheranim \vee bird$ $featheranim \rightarrowtail bird$ $bird \rightarrowtail featheranim$ $penguin \rightarrowtail \neg fly$ $bird \rightarrowtail fly$ $penguin \wedge featheranim \wedge bird \rightarrowtail \neg fly$	BCOi	30/30	16.76	30/30	32.67	30/30	8.83
	FME	MEM	–	MEM	–	MEM	–
$penguin \rightarrowtail featheranim \vee bird$ $featheranim \rightarrowtail bird$ $bird \rightarrowtail featheranim$ $penguin \rightarrowtail \neg fly$ $bird \rightarrowtail fly$ $penguin \wedge featheranim \wedge bird \rightarrowtail fly$	BCOi	30/30	16.76	30/30	43.73	30/30	16.27
	FME	MEM	–	MEM	–	MEM	–

Табела 6.6: Резултати тестирања за примере код којих $\Delta \not\vdash \alpha \rightarrowtail \beta$ и $\Delta \not\vdash \alpha \rightarrowtail \neg\beta$

Пример	Метода	Δ		Φ_1		Φ_2	
		Статус	Време	Статус	Време	Статус	Време
$quaker \rightarrowtail pacifist$ $republican \rightarrowtail \neg pacifist$ $quaker \wedge republican \rightarrowtail pacifist$	BCOi	30/30	2.24	30/30	0.83	30/30	1.99
	FME	sat	9866.65	sat	10620.05	sat	25394.76
$quaker \rightarrowtail pacifist$ $republican \rightarrowtail \neg pacifist$ $quaker \wedge republican \rightarrowtail \neg pacifist$	BCOi	30/30	2.24	30/30	0.62	30/30	1.50
	FME	sat	9866.65	sat	29637.52	sat	10675.40
$quaker \rightarrowtail pacifist$ $republican \rightarrowtail \neg pacifist$ $ecologist \rightarrowtail pacifist$ $quaker \wedge ecologist \wedge republican \rightarrowtail pacifist$	BCOi	30/30	12.02	30/30	11.92	30/30	9.08
	FME	MEM	–	MEM	–	MEM	–
$quaker \rightarrowtail pacifist$ $republican \rightarrowtail \neg pacifist$ $ecologist \rightarrowtail pacifist$ $quaker \wedge ecologist \wedge republican \rightarrowtail \neg pacifist$	BCOi	30/30	12.02	30/30	11.13	30/30	9.46
	FME	MEM	–	MEM	–	MEM	–
$teenAger \rightarrowtail poor$ $teenAger \rightarrowtail student$ $poor \rightarrowtail employed$ $student \rightarrowtail \neg employed$ $teenAger \rightarrowtail employed$	BCOi	30/30	18.04	30/30	23.64	30/30	11.98
	FME	MEM	–	MEM	–	MEM	–
$teenAger \rightarrowtail poor$ $teenAger \rightarrowtail student$ $poor \rightarrowtail employed$ $student \rightarrowtail \neg employed$ $teenAger \rightarrowtail \neg employed$	BCOi	30/30	18.04	30/30	18.23	30/30	20.24
	FME	MEM	–	MEM	–	MEM	–
$young \rightarrowtail lawAbiding$ $joyRiders \rightarrowtail \neg lawAbiding$ $joyRiders \rightarrowtail young$ $joung \wedge joyRiders \rightarrowtail lawAbiding$	BCOi	30/30	12.77	30/30	14.62	30/30	4.02
	FME	sat	84695.65	MEM	–	MEM	–
$young \rightarrowtail lawAbiding$ $joyRiders \rightarrowtail \neg lawAbiding$ $joyRiders \rightarrowtail young$ $joung \wedge joyRiders \rightarrowtail \neg lawAbiding$	BCOi	30/30	12.77	30/30	17.47	30/30	6.56
	FME	sat	84695.65	MEM	–	MEM	–

Табела 6.7: Резултати добијени применом FME методе за примере када $\Delta \mid\!\sim \alpha \rightarrowtail \beta$

Пример	Статус	Време	Пример	Статус	Време
$bird \rightarrowtail fly$ $penguin \rightarrowtail bird$ $penguin \rightarrowtail \neg fly$ $bird \wedge penguin \rightarrowtail \neg fly$	un-sat	626.75	$bird \rightarrowtail fly$ $penguin \rightarrowtail bird$ $penguin \rightarrowtail \neg fly$ $fly \rightarrowtail \neg penguin$	un-sat	0.01
$bird \rightarrowtail fly$ $penguin \rightarrowtail bird$ $penguin \rightarrowtail \neg fly$ $bird \rightarrowtail \neg penguin$	un-sat	0.01	$bird \rightarrowtail fly$ $penguin \rightarrowtail bird$ $penguin \rightarrowtail \neg fly$ $bird \vee penguin \rightarrowtail \neg penguin$	un-sat	0.01
$bird \rightarrowtail fly$ $penguin \rightarrowtail bird$ $penguin \rightarrowtail \neg fly$ $bird \vee penguin \rightarrowtail fly$	un-sat	0.01	$\theta \rightarrowtail \phi$ $\theta \rightarrowtail \psi$ $\theta \rightarrowtail \phi \wedge \psi$	MEM	–
$\theta \rightarrowtail \phi$ $\phi \rightarrowtail \psi$ $\theta \vee \phi \rightarrowtail \psi$	un-sat	0.01	$\theta \rightarrowtail \psi$ $\theta \rightarrowtail \phi$ $\theta \wedge \phi \rightarrowtail \psi$	MEM	–
$\theta \rightarrowtail \phi$ $\theta \wedge \phi \rightarrowtail \psi$ $\theta \rightarrowtail \psi$	un-sat	0.01	$\theta \rightarrowtail \phi$ $\theta \wedge \phi \rightarrowtail \psi$ $\theta \rightarrowtail \psi$	un-sat	0.01
$\theta \rightarrowtail \phi$ $\phi \rightarrowtail \theta$ $\theta \rightarrowtail \psi$ $\phi \rightarrowtail \psi$	un-sat	0.01	$\theta \rightarrowtail \phi \rightarrow \psi$ $\theta \rightarrowtail \phi$ $\theta \rightarrowtail \psi$	MEM	–

Глава 7

Закључна разматрања

Главни део представљен у овом раду се односи на развој и тестирање хеуристике засноване на BCOi методи за решавање CPSAT-ε проблема и проблема дифолтног закључивања. За CPSAT-ε проблем за вероватносну логику, представљену у раду [101], није постојао аутоматски доказивач теорема. Иако се овај проблем може представити у облику проблема линеарног програмирања, број непознатих и број неједнакости у добијеном систему расте експоненцијално са повећањем броја исказних слова у посматраној формули. Додатно, применом егзактне (директне) методе за решавање добијеног система неједнакости, попут FME методе, већина проблема постаје практично нерешива, због новог пораста броја неједнакости у систему након сваке елиминације непознате. Оваква ситуација сугерише неопходност хеуристичког приступа решавању проблема. У овом раду изабрана је BCO метода, прецизније, њена варијанта са поправком BCOi, која је до сада, углавном, коришћена за решавање комбинаторних проблема код којих постоји неко решење, а циљ је био то решење оптимизовати. Предложени приступ представља прву примену BCOi засноване хеуристике на проблем који, не само да захтева проналажење решења, већ и потврду постојања решења. Разлог за избор ове методе је тај што се у разним истраживањима показало да методе засноване на популацији, углавном, дају боље резултате од метода заснованих на локалном претраживању. На основу природе CPSAT-ε проблема, свака пчела је у свом лету унапред имала три важна корака:

(i) да пронађе бољи скуп атома који имају вероватноће веће од нула;

(ii) да поправи вероватноће које су додељене атомима да би се приближила траженом решењу;

(iii) да провери да ли је решење пронађено.

Представљени резулати показују велику надмоћ BCOi методе над приступом базираном на FME методи. Доказивачи теорема који су постојали до сада, решавали су PSAT проблеме у којима су фигурисали оператори апсолутне вероватноће, за

разлику од оператора условне вероватноће који је овде разматран, и посматране вероватноће су имале реалне вредности, док се сада први пут разматрају вероватноће из Хардијевог поља бројева. Обзиром да ово представља први хеуристички приступ решавању овог проблема, није било могуће извршити поређење добијених резултата ни са чим другим до са директним приступом. Са друге стране, осим што је обезбеђен први практични алат за решавање CPSAT-ε проблема, такође је дат и скуп примера који се могу користити у оцени будућих приступа.

Имајући у виду начин рада општег BCOi алгоритма, било је могуће направити дистрибуирани доказивач којим се добија знатно убрзање. Свака итерација BCOi алгоритма се одвија независно, па њихово паралелно извршавање битно смањује потребно време, што је било и очекивано.

Представљен приступ за решавање CPSAT-ε проблема отворио је могућност за примену ове методе у дифолтном закључивању. Дифолтни приступ у представљању знања и закључивању представља врло битан метод, обзиром да подржава закључивање са непотпуним информацијама. Немонотоно закључивање се може користити на многим местима. Један од најочигледнијих примера је медицинска дијагностика, где резултати нових анализа могу бити у контрадикцији са усвојеним закључком. Слична ситуација је и у закључивању у односу на правне регулативе и прописе, у спецификацији система и софтвера и тако даље. Дифолтна логика је, могуће, најпопуларнији метод немонотоног резоновања, пре свега због једноставности записа дифолта.

У дифолтном резоновању, до сада није коришћен хеуристички приступ. Обзиром на везу дифолтне логике и $LPCP$ логике, и овде је било могуће посматрани проблем представити у облику проблема линеарног програмирања. Као и у случају решавања CPSAT-ε проблема, директан приступ коришћењем FME методе није могао да да задовољавајуће резултате. Иако је на први поглед деловало да ће једноставна примена већ развијеног доказивача за CPSAT-ε проблем дати задовољавајуће резултате, показало се да специфичности добијених формула захтевају додатна подешавања, како у коришћеним стратегијама приликом модификације решења, тако и у начину представљања вредности из Хардијевог поља бројева. Општи алгоритам је остао исти као при решавању CPSAT-ε проблема, али је поред хеуристичког приступа у модификацији решења, додата и Нелдер-Мид метода и урађена је детаљнија анализа резултата добијених комбиновањем ових метода. Такође, поређени су приступи када се за сваку пчелу генерише ново иницијално решење на почетку сваке итерације и ситуација када се преноси део решења из претходне итерације, што се показало доста ефикаснијим приступом. Ефикасност је додатно повећана и када је дозвољено да се вредност функције циља поквари током модификације решења, а не да се увек задржава боље решење, како је то био случај приликом решавања CPSAT-ε проблема. Иако развијена метода није комплетна, висок проценат успешности и комбинација са FME методом примењеном на делове добијених система неједнакости, даје добре смернице за извођење закључака у дифолтном резоновању.

Понуђен метод отвара могућност за решавање PSAT проблема за друге сличне логике (попут логика представљених у радовима [23, 84, 88, 100]). Додатним тестирањима различитих стратегија могли би се добити бољи резулати приликом решавања CPSAT-ε проблема, као што је то учињено код дифолтног закључивања. Са друге стране, без обзира на преношење дела решења из једне итерације у другу приликом дифолтног резоновања, и у таквом приступу би се могао развити дистибуирани доказивач теорема у циљу повећања ефикасности. Обзиром да је ово први хеуристички приступ у решавању ових проблема, од интереса би било развити и друге хеуристичке методе и извршити поређење резултата добијених применом различитих хеуристика.

Додатак А

А.1 Фурије-Моцкин метода елиминације

Једна од директних метода за решавање система линеарних неједнакости је Фурије-Моцкин метода елиминације. Ова метода је развијена за решавање у Еуклидовом, реалновредносном простору, а за потребе овог рада решења система се траже у не-Архимедовом, хиперреалном простору. Посматра се систем од m линеарних неједнакости у n-димензионом простору. Посматрани систем се може записати у облику

$$\mathbf{A}x \leqslant \mathbf{b} \tag{А.1.1}$$

где је $\mathbf{A}$ матрица коефицијената из $\mathbb{R}^{m\times n}$, $x = (x_1, \ldots, x_n)^{\mathrm{T}} \in \mathbb{R}^n$ вектор непознатих и $\mathbf{b}$ колона вектор из $\mathbb{R}^n$. Нека је

$$X = \{x \in \mathbb{R}^n : \mathbf{A}x \leqslant \mathbf{b}\} \tag{А.1.2}$$

скуп решења ситема и нека $X^{[k]}$ означава пројекцију скупа X на простор разапет на последњих $n-k$ координата:

$$X^{[k]} = \left\{(x_{k+1}, \ldots, x_n) \in \mathbb{R}^{n-k} : \exists (x_1, \ldots, x_k) \in \mathbb{R}^k, (x_1, \ldots, x_n) \in X\right\}$$

Фурије-Моцкин метода ([12, 14, 25, 57, 61, 81, 106]) сукцесивно елиминише непознате $(x_1, \ldots, x_{n-1})$ из ситема (А.1.1) и израчунава матрице $A^{[k]}$ и вектор $b^{[k]}$ тако да

$$X^{[k]} = \left\{x^{[k]} \in \mathbb{R}^{n-k} : A^{[k]}x^{[k]} \leqslant b^{[k]}\right\}, k = 1, \ldots, n-1$$

где је $x^{[k]} = (x_{k+1}, \ldots, x_n)^{\mathrm{T}}$.

У циљу елиминације непознате x_1, свака од m неједнакости из система (А.1.1) се множи одговарајућом позитивном вредношћу да би вредности у првој колони матрице $\mathbf{A}$ постале ± 1 или 0. Без губитка општости, може се претпоставити да

се оригиналан систем може записати у облику

$$\begin{aligned} +1 \cdot x_1 + \alpha_i(x^{[1]}) &\leqslant 0, \quad i \in M_+, \\ -1 \cdot x_1 + \alpha_i(x^{[1]}) &\leqslant 0, \quad i \in M_-, \\ 0 \cdot x_1 + \alpha_i(x^{[1]}) &\leqslant 0, \quad i \in M_0, \end{aligned}$$

где је $\alpha_i(x^{[1]}) = \alpha_{i2}x_2 + \cdots + \alpha_{in}x_n + \beta_i$ дати афини облик за $x^{[1]} = (x_2, \ldots, x_n)^{\mathrm{T}} \in \mathbb{R}^{n-1}$ и M_+, M_-, M_0 дисјунктни скупови индекса неједнакости система (А.1.1) при чему важи да је

$$M_+ \cup M_- \cup M_0 = \{1, \ldots, m\}.$$

Лако се види да за свако фиксирано $x^{[1]}$, неједнакост са индексом $i \in M_+ \cup M_-$ може бити задовољена за неки реалан број x_1 ако и само ако свака горња граница $-\alpha_i(x^{[1]}), i \in M_+$ за x_1 је већа од сваке доње границе $\alpha_j(x^{[1]}), j \in M_-$, тј.

$$-\alpha_i(x^{[1]}) \geqslant \alpha_j(x^{[1]}), \qquad i \in M_+, \quad j \in M_-$$

Комбиновањем ових $|M_+| \cdot |M_-|$ неједнакости са преосталих $|M_0|$ неједнакости из система (А.1.1) које не садрже x_1 добија се систем од $|M_+| \cdot |M_-| + |M_0|$ линеарних неједнакости

$$\begin{aligned} &\alpha_i(x^{[1]} + \alpha_j(x^{[1]}) \leqslant 0, \qquad (i,j) \in M_+ \times M_- \\ &\alpha_i(x^{[1]}) \leqslant 0, \qquad i \in M_0 \end{aligned}$$

чије решење је скуп $X^{[1]}$. Претходни систем може да се напише у облику

$$A^{[1]}x^{[1]} \leqslant b^{[1]}$$

са одговарајућом матрицом $A^{[1]}$ и вектором $b^{[1]}$. Овиме се дефинише

$$X^{[1]} = \left\{ x^{[1]} \in \mathbb{R}^{n-1} : A^{[1]}x^{[1]} \leqslant b^{[1]} \right\}.$$

Слично се елиминацијом непознате x_2 из система $A^{[2]}x^{[2]} \leqslant b^{[2]}$ добија опис друге пројекције $X^{[2]} = \left\{ x^{[2]} \in \mathbb{R}^{n-2} : A^{[2]}x^{[2]} \leqslant b^{[2]} \right\}$ и тако даље. Након $n-1$ корака описаном процедуром добија се $n-1$ матрица $A^{[k]}$ и вектор $b^{[k]}$ такви да је $X^{[k]} = \left\{ x^{[k]} \in \mathbb{R}^{n-k} : A^{[k]}x^{[k]} \leqslant b^{[k]} \right\}, k = 1, \ldots, n-1$.

А.1.1 Решење система линеарних неједнакости и проблем линеарног програмирања

Ако је скуп решења $X = \{x \in \mathbb{R}^n : \mathbf{A}x \leqslant \mathbf{b}\}$ непразан, тада су непразне и пројекције $X^{[k]} \subseteq \mathbb{R}^{n-k}$, $k = 1, \ldots, n-1$ и обратно. Прецизније речено, ако $\mathbf{A}x \leqslant \mathbf{b}$ има решење, тада је

$$X^{[n-1]} = \left\{ x^{[n-1]} \in \mathbb{R} : A^{[n-1]}x^{[n-1]} \leqslant b^{[n-1]} \right\}.$$

непразан интервал за непознату $x^{[n-1]} = x_n$. За дато $A^{[n-1]}$ и $b^{[n-1]}$ лако се може наћи тачка таква да $\overline{x}_n \in X^{[n-1]}$. Тада се заменом $x_n = \overline{x}_n$ у систему $A^{[n-2]}x^{[n-2]} \leqslant b^{[n-2]}$ добија нови решиви систем линеарних неједнакости чији је скуп решења интервал $\{x_2 \in \mathbb{R} : (x_n - 1, \overline{x}_n) \in X^{[n-2]}\}$. Решавањем овог система који садржи једну непознату добија се тачка $\overline{x}^{[n-2]} = (\overline{x}_{n-1}, \overline{x}_n) \in X^{[n-2]}$, која се може заменити у систему $A^{[n-3]}x^{[n-3]} \leqslant b^{[n-3]}$ и тако даље. Понављањем ове процедуре замене уназад, Фурије-Моцкин методом се може одредити решење $(\overline{x}_1, \ldots, \overline{x}_n)$ ма ког задовољивог система линеарних неједнакости $\mathbf{A}x \leqslant \mathbf{b}$. Историјски, ово представља метод "пред-линеарног програмирања" за решавање система линеарних неједнакости[25, 106].

А.2 Нелдер-Мид метода оптимизације

Секвенцијалну симплекс методу су предложили Спендлеј (енг. *W. Spendley*), Хекст (енг. *G. R. Hext*) и Химсворт (енг. *F. R. Himsworth*) [110], а читаву идеју су даље проширили Нелдер (енг. *J. A. Nelder*) и Мид (енг. *R. Mead*) [83]. Метода је развијена за решавање оптимизационог проблема типа

$$\min_x f(x)$$

при чему је f функција од n променљивих, без ограничења.

Нека су P_0, P_1, ..., P_n тачке димензије n које дефинишу тренутни симплекс. Биће коришћна ознака y_i за $f(P_i)$, тако да

- индекс h је такав да је $y_h = \max_i(y_i)$
- индекс l је такав да је $y_l = \min_i(y_i)$

Са $\overline{P}$ ће бити означен центроид, тј. тачка подједнако удаљена од свих тачака P_i, $i \neq h$, а са $[P_iP_j]$ растојање између тачака P_i и P_j. У свакој фази P_h се замењује новом вредношћу и користе се три операције: *рефлексија, контракција и експанзија.*

Рефлексија тачке P_h се означава са P^* и њене координате су дефинисане релацијом

$$P^* = (1 + \alpha)\overline{P} - \alpha P_h$$

где је α позитивна константа, коефицијент рефлексије. Ако је вредност y^* између вредности y_h и y_l, тада се P_h замењује са P^* и процес почиње поново са новим симплексом.

Ако је $y^* < y_l$, тј. ако је рефлексија дала нови минимум, тада се P^* проширује са P^{**} релацијом

$$P^{**} = \gamma P^* + (1 - \gamma)\overline{P}$$

Коефицијент **експанзије** γ је већи од 1 и представља однос растојања $[P^{**}\overline{P}]$ и $[P^{*}\overline{P}]$. Ако је $y^{**} < y_l$, тачка P_h се замењује тачком P^{**} и процес почиње поново; али ако је $y^{**} > y_l$, тада експанзија није успела и P_h се замењује са P^* пре поновног почетка.

Ако се рефлексијом P на P^* добије да је $y^* > y_i$ за свако $i \neq h$, тј. да се заменом P са P^* добија да је y^* максимум, тада се дефинише ново P_h које је или старо P_h или P^* у зависности за које y има мању вредност и формира се

$$P^{**} = \beta P_h + (1 - \beta)\overline{P}.$$

Коефицијент **контракције** β има вредност између 0 и 1 и представља однос растојања $[P^{**}\overline{P}]$ и $P\overline{P}$. Вредност P^{**} се узима за P_h и процес креће поново, осим уколико је $y^{**} > \min(y_h, y^*)$, тј. тачка контракције је гора и од P_h и од P^*. Ако контракција не успе, све вредности P_i се мењају са $(P_i + P_l)/2$ и процес почиње из почетка. Као критеријум за заустављање се може узети ситуација када просечна вредност тренутних y_i постане довољно мала.

А.3 Паралелизација извршавања програма коришћењем MPI методе

Брже извршавање компликованих израчунавања, за којима данас, са развојем науке, постоји све већа потреба, захтевало је развој различитих метода паралелизације процеса. Један од тренутно најпопуларнијих метода је стандардно окружење за размену порука MPI (енг. *Message Passing Interface*). MPI представља спецификацију стандардне библиотеке функција за паралелни модел прослеђивања порука. Приликом формирања, MPI стандард је преузео конструкције разних постојећих система који су се добро показали у пракси и уградио их у један поуздан и ефикасан систем.

Како је MPI стандард независан од платформе и оперативног система на којима се извршава, свака MPI имплементација мора додатно да обезбеди улазно/излазне операције и начин извршавања апликација на датом конкретном оперативном систему.

MPI поседује висок степен флексибилности, чему иде у прилог чињеница да се MPI систем може извршавати на мрежи хетерогених рачунара, односно скупу процесора различитих архитектура. Корисник не мора да води рачуна о томе да ли се поруке размењују између процесора истих или различитих архитектура, јер MPI аутоматски обавља све неопходне конверзије података и обезбеђује одговарајући протокол за комуникацију.

MPI програм може да се састоји из више примерака секвенцијалног програма који комуницирају позивима одговарајућих функција MPI библиотеке. Ове функције се могу сврстати у следеће групе:

1. Функције које врше иницијализацију, управљање и окончавање комуникације;

2. Функције које служе за комуникацију између парова процеса;
3. Функције које изводе операције над групом процеса;
4. Функције за креирање произвољних типова података.

Помоћу наведених MPI функција, на различитим типовима архитектура паралелних рачунара, реализују се следећи процеси:

1. Навођењем директива оперативни систем распоређује копије извршног програма на сваки чвор кластера (процесоре);
2. Сваки процесор извршава своју копију програма;
3. Различити процесори парелелно извршавају онај део програма у коме се користи идентификациони број процеса.

У наставку ће бити направљен кратак преглед функција MPI библиотека које се најчешће користе приликом генерисања паралелних програма [34, 35, 98].

Функција MPI_Init

Сваки MPI процес прво позива функцију MPI_Init. Она обезбеђује да систем изврши подешавања потребна за позиве осталих MPI функција. Позив ове функције не мора да буде прва наредба програм, чак не мора да буде смештен у main функцији C програма, али се мора навести пре позива осталих MPI функција. Синтакса ове функције у програмском језику C је

```
MPI_Init(&argc,&argv);
```

Функције MPI_Comm_rank и MPI_Comm_size

Након MPI иницијализације, сваки активан процес постаје члан комуникатора MPI_COMM_WORLD, стандардног комуникатора MPI библиотеке. Комуникатор је апстрактни објекат који се брине о окружењу за размену порука међу процесима. Уколико је потребно, корисник може сам креирати комуникаторе дељењем процеса на комуникаторске групе, при чему мора узети у обзир да процеси могу комуницирати само ако су унутар исте групе.

Процеси унутар комуникатора су уређени по неком редоследу. У комуникатору са p процеса, сваки процес добија јединствени број id између 0 и $p-1$. На основу id вредности процеси могу да утврде за који део израчунавања и/или података су задужени.

Позивом функције MPI_Comm_rank се сваком процесу додељује јединствени број, а функцијом MPI_Comm_size се одређује укупан број процеса у комуникатору. Синтакса ових функција у програмском језику C је

```
MPI_Comm_rank(MPI_COMM_WORLD,&id);
MPI_Comm_size(MPI_COMM_WORLD,&p);
```

Уколико је један процес у две комуникаторске групе, његов id ће за сваку групу бити другачији.

Функција MPI_Bcast

Функција MPI_Bcast омогућава да процес проследи један или више података истог типа свим процесима у комуникатору. Синтакса ове функције у програмском језику C је

```
int MPI_Bcast(
    void *buffer,
    int count,
    MPI_Datatype datatype,
    int root,
    MPI_Comm comm
);
```

где први аргумент buffer садржи адресу првог податка који треба проследити, аргумент count означава број података који се прослеђују. Функција подразумева да су сви подаци истог типа datatype и да се налазе у меморији један за другим. Параметар root представља id процеса који шаље податка, док параметар comm означава комуникатор коме припадају процеси који учествују у тренутној комуникацији.

Функција MPI_Send

Приликом извршавања програма, често је неопходно обезбедити комуникацију између два конкретна процеса. Ова комуникација може да се обави разменом порука. Процес који шаље поруку позива функцију MPI_Send чије је C синтакса дата са

```
int MPI_Send(
    void *message,
    int count,
    MPI_Datatype datatype,
    int dest,
    int tag,
    MPI_Comm comm
);
```

Први параметар message представља почетну адресу податка који ће бити послат. Други параметар count представља број података који се шаљу, док параметар datatype предствља њихов тип. Параметар dest предствља id процеса који треба да прими податке. Помоћу параметра tag може се обезбедити идентификација сврхе поруке. Параметар comm, и у овом случају, означава комуникатор.

Функција MPI_Recv

Процес који треба да прими поруку, коју шаље други процес, позива функцију MPI_Recv. Синтакса ове функције је, донекле, слична синтакси функције MPI_Send:

```
int MPI_Recv(
    void *message,
    int count,
    MPI_Datatype datatype,
    int source,
    int tag,
    MPI_Comm comm,
    MPI_Status *status
);
```

са тим што сада четврти параметар source означава id процеса који шаље поруку. Пре позива функције MPI_Recv процес мора да алоцира структуру типа MPI_Status, а параметар status садржи показивач на ову структуру. Ова структура садржи податке о томе који процес је послао поруку, која је сврха поруке и информације о могућој грешци која је настала.

Функција MPI_Finalize

Након што сви процеси заврше све своје MPI позиве, позива се функција MPI_Finalize која омогућава систему да ослободи све ресурсе који су били заузети од стране MPI програма.

Библиографија

[1] E. W. Adams. *The logic of conditionals: An application of probability to deductive logic*. Springer, 1975.

[2] S. Aljančić. *Uvod u realnu i funkcionalnu analizu*. Gradjevinska knjiga, Beograd, 1968.

[3] A. Alsheddy. *Empowerment scheduling: a multi-objective optimization approach using Guided Local Search*. PhD thesis, University of Essex, 2011.

[4] T. Bayes. *An essay towards solving a problem in the doctrine of chances*. 1763.

[5] S. Benferhat, A. Saffiotti, P. Smets. Belief functions and default reasoning. *Artificial Intelligence*, 122(1):1–69, 2000.

[6] B. Bolzano. *Wissenschaftslehre*. 1837.

[7] G. Boole. *The mathematical analysis of logic*. Philosophical Library, 1847.

[8] G. Boole. *An investigation of the laws of thought: on which are founded the mathematical theories of logic and probabilities*. Dover Publications, 1854.

[9] R. Booth, J. B. Paris. A note on the rational closure of knowledge bases with both positive and negative knowledge. *Journal of Logic, Language and Information*, 7(2):165–190, 1998.

[10] S. Camazine, J. Sneyd. A model of collective nectar source by honey bees: self-organization through simple rules. *Journal of Theoretical Biology*, 149:547–571, 1991.

[11] R. Chiong, editor. *Nature-inspired algorithms for optimization*. Springer, 2009.

[12] V. Chvatal. *Linear programming*. W. H. Freeman and Company, 1983.

[13] D. Cvetković, M. Čangalović, Dj. Dugošija, V. Kovačević-Vujčić, S. Simić, J. Vuleta. *Kombinatorna optimizacija (Matematička teorija i algoritmi)*. DOPIS, Beograd, 1996.

[14] G. B. Dantzig, B. C. Eaves. Fourier-Motzkin elimination and its dual. *J. Comb. Theory, Ser. A*, 14(3):288–297, 1973.

[15] T. Davidović. *Rasporedjivanje zadataka na višeprocesorske sisteme primenom metaheuristika*. PhD thesis, Matematički fakultet Univerziteta u Beogradu, Feb. 2006.

[16] T. Davidović, D. Ramljak, M. Šelmić, D. Teodorović. Bee colony optimization for the p-center problem. *Computers and Operations Research*, 38(10):1367–1376, 2011.

[17] T. Davidović, D. Teodorović, M. Šelmić. Bee colony optimization part I: The algorithm overview. *Yugoslav Journal of Operational Research*, 25(1):33–56, 2015.

[18] A. De Morgan. *Formal logic*. 1847.

[19] J. Dix, U. Furbach, I. Niemelä. Nonmonotonic reasoning: Towards efficient calculi and implementations. *Handbook of Automated Reasoning*, 2(18):1121–1234, 2001.

[20] M. Djeffal, H. Drias. Multilevel Bee Swarm Optimization for Large Satisfiability Problem Instances. *Intelligent Data Engineering and Automated Learning–IDEAL 2013*, pages 594–602, 2013.

[21] M. Dorigo, M. Birattari. Ant colony optimization. In *Encyclopedia of machine learning*, pages 36–39. Springer, 2010.

[22] T. Eiter, T. Lukasiewicz. Default reasoning from conditional knowledge bases: Complexity and tractable cases. *Artificial Intelligence*, 124(2):169–241, 2000.

[23] R. Fagin, J. Y. Halpern, N. Megiddo. A logic for reasoning about probabilities. *Information and computation*, 87(1):78–128, 1990.

[24] T. A. Feo, M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995.

[25] C. A. Floudas, P. M. Pardalos. *Encyclopedia of optimization*, volume 1. Springer, 2008.

[26] H. Gaifman. Concerning measures in first order calculi. *Israel journal of mathematics*, 2(1):1–18, 1964.

[27] H. Gaifman, M. Snir. Probabilities over rich languages, testing and randomness. *The journal of symbolic logic*, 47(03):495–548, 1982.

[28] G. Georgakopoulos, D. Kavvadias, C. H. Papadimitriou. Probabilistic satisfiability. *Journal of Complexity*, 4(1):1 – 11, 1988.

[29] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986.

[30] F. Glover, M. Laguna. *Tabu search*. Springer, 1999.

[31] K. Gödel. *Über die Vollstandigkeit des Logikkalkulus*. PhD thesis, University of Vienna. 1929.

[32] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publ. Comp., Inc., 1989.

[33] M. Goldszmidt, J. Pearl. On the consistency of defeasible databases. *Artificial Intelligence*, 52(2):121–149, 1991.

[34] W. Gropp, E. Lusk. *Users Guide for* `mpich` *a Portable Implementation of MPI.* University of Chicago, Argonne National Laboratory, 1996.

[35] W. Gropp, E. Lusk, A. Skjellum. *Using MPI: Portable Parallel Programming with the Message–Passing Interface.* The MIT Press, 1994.

[36] T. Hailperin. *Boole's logic and probability.* North Holland Amsterdam, 1976.

[37] T. Hailperin et al. Probability logic. *Notre Dame Journal of Formal Logic*, 25(3):198–212, 1984.

[38] J. Y. Halpern. Lexicographic probability, conditional probability, and nonstandard probability. In *Proceedings of the 8th conference on Theoretical aspects of rationality and knowledge*, pages 17–30. Morgan Kaufmann Publishers Inc., 2001.

[39] P. J. Hammond. Non-archimedean subjective probabilities in decision theory and games. *Mathematical Social Sciences*, 38(2):139–156, 1999.

[40] P. Hansen, B. Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, 44(4):279–303, 1990.

[41] P. Hansen, B. Jaumard. Probabilistic satisfiability. In *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, pages 321–367. Springer, 2000.

[42] P. Hansen, N. Mladenović. An Introduction to Variable Neighborhood Search. In Voss, S. *et al.*, editor, *Metaheuristics, Advances and Trends in Local Search Paradigms for Optimization*, pages 433–458. Kluwer, Dordrecht, 1999.

[43] P. Hansen, N. Mladenović. Fundamentals of Variable Neighborhood Search. In *Proc. 10. Congr. Yugoslav Mathematicians*, pages 57–72, Beograd, 2001.

[44] P. Hansen, N. Mladenović. Variable Neighborhood Search: Principles and Applications. *Europ. J. Oper. Res.*, 130:449–467, 2001.

[45] P. Hansen, N. Mladenović. Developments of the Variable Neighborhood Search. In C. Ribeiro, P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 415–439. Kluwer Academic Publishers, 2002.

[46] P. Hansen, N. Mladenović. Variable Neighbourhood Search. In F. Glover, G. Kochenagen, editors, *Handbook of Metaheuristics*, pages 145–184. Kluwer Academic Publishers, Dordrecht, 2003.

[47] A. Hertz, E. Taillard, D. de Werra. Tabu search. In E. Aarts, J. K. Lenstra, editors, *Local Search in Combinatorial optimization*, pages 121–136. John Wiley & Sons Ltd., 1997.

[48] J. H. Holland. *Adaptation in Natural and Artificial Systems.* Ann Arbor: The University of Michigan Press, 1975.

[49] D. N. Hoover. Probability logic. *Annals of mathematical logic*, 14(3):287–313, 1978.

[50] D. Jovanović, N. Mladenović, Z. Ognjanović. Variable neighborhood search for the probabilistic satisfiability problem. *Metaheuristics Progress in Complex Systems Optimization*, pages 173–188, 2007.

[51] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Erciyes University, Engineering Faculty Computer Engineering Department Kayseri/Turkiye, 2005.

[52] D. Kavvadias, C. H. Papadimitriou. A linear programming approach to reasoning about probabilities. *Annals of Mathematics and Artificial Intelligence*, 1(1-4):189–205, 1990.

[53] H. J. Keisler. Hyperfinite model theory. *Logic Colloquium*, 76:5–110, 1977.

[54] H. J. Keisler. *Elementary calculus: an infinitesimal approach.* Prindle Weber & Schmidt, 1986.

[55] Y. Kilani. Comparing the performance of the genetic and local search algorithms for solving the satisfiability problems. *Applied Soft Computing*, 10(1):198–207, 2010.

[56] S. Kirkpatrick, C. D. Gelatt Jr, M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[57] D. A. Kohler. Translation of a report by Fourier on his work on linear inequalities. *Opsearch*, 10:38–42, 1973.

[58] A. Kolmogorov. *Grundbegriffe der Wahrscheinlichkeitsrechnung.* 1933.

[59] S. Kraus, D. Lehmann, M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial intelligence*, 44(1):167–207, 1990.

[60] A. Kron. *Odnos polivalentnih logika i teorije verovatnoće.* Naučna Knjiga, 1967.

[61] H. W. Kuhn. Solvability and consistency for linear equations and inequalities. *American Mathematical Monthly*, pages 217–232, 1956.

[62] J. H. Lambert. *Neues Organon oder Gedanken über die Erforschung und Bezeichnung des Wahren und dessen Unterscheidung vom Irrthum und Schein*, volume 2. Wendler, 1764.

[63] D. Lehmann, M. Magidor. What does a conditional knowledge base entail? *Artificial Intelligence*, 55(1):1–60, 1992.

[64] G. W. Leibnitz. *De Conditionibus.* 1665.

[65] G. W. Leibnitz. *Specimen juris.* 1669.

[66] G. W. Leibnitz. *De Nouveaux essais.* 1765.

[67] H. R. Lourenço, O. C. Martin, T. Stützle. Iterated local search: Framework and applications. In *Handbook of Metaheuristics*, pages 363–397. Springer, 2010.

[68] Z. Lü, J.-K. Hao. Adaptive memory-based local search for MAX-SAT. *Applied Soft Computing*, 12(8):2063–2071, 2012.

[69] T. Lukasiewicz. Probabilistic default reasoning with conditional constraints. *Annals of Mathematics and Artificial Intelligence*, 34(1-3):35–88, 2002.

[70] T. Lukasiewicz. Weak nonmonotonic probabilistic logics. *Artificial Intelligence*, 168(1):119–161, 2005.

[71] T. Lukasiewicz. Nonmonotonic probabilistic logics under variable-strength inheritance with overriding: Complexity, algorithms, and implementation. *International Journal of Approximate Reasoning*, 44(3):301–321, 2007.

[72] P. Lučić, D. Teodorović. Bee system: modeling combinatorial optimization transportation engineering problems by swarm intelligence. In *Preprints of the TRISTAN IV Triennial Symposium on Transportation Analysis*, pages 441–445. Sao Miguel, Azores Islands, 2001.

[73] P. Lučić, D. Teodorović. Transportation modeling: an artificial life approach. In *Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence*, pages 216–223, Washington, DC, 2002.

[74] P. Lučić, D. Teodorović. Computing with bees: attacking complex transportation engineering problems. *International Journal on Artificial Intelligence Tools*, 12:375–394, 2003.

[75] H. MacColl. Symbolic reasoning. *Mind*, pages 493–510, 1897.

[76] P. Maksimović, T. Davidović. Parameter calibration in the bee colony optimization algorithm. In *XI Balcan Conference on Operational Research*, pages 263–272, BALCOR 2013, Beograd-Zlatibor, Serbia, 2013.

[77] E. Marchiori, C. Rossi. A flipping genetic algorithm for hard 3-SAT problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 393–400, 1999.

[78] G. Marković, D. Teodorović, V. Aćimović-Raspopović. Routing and wavelength assignment in all-optical networks based on the bee colony optimization. *AI Commun.*, 20(4):273–285, 2007.

[79] H. McColl. The calculus of equivalent statements and integration limits. *Proceedings of the London Mathematical Society*, 1(1):9–20, 1877.

[80] N. Mladenović, P. Hansen. Variable Neighborhood Search. *Comput. & OR*, 24(11):1097–1100, 1997.

[81] T. S. Motzkin. *Beiträge zur Theorie der linearen Ungleichungen.* Azriel, 1936.

[82] A. Munawar, M. Wahib, M. Munetomo, K. Akama. Hybrid of genetic algorithm and local search to solve max-sat problem using nvidia cuda framework. *Genetic Programming and Evolvable Machines*, 10(4):391–415, 2009.

[83] J. A. Nelder, R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[84] N. J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–87, 1986.

[85] Z. Ognjanović, J. Kratica, M. Milovanović. A genetic algorithm for satisfiability problem in a probabilistic logic: A first report. *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 805–816, 2001.

[86] Z. Ognjanović, U. Midić, J. Kratica. A genetic algorithm for probabilistic SAT problem. *Artificial Intelligence and Soft Computing-ICAISC 2004*, pages 462–467, 2004.

[87] Z. Ognjanović, U. Midić, N. Mladenović. A hybrid genetic and variable neighborhood descent for probabilistic SAT problem. In *Hybrid Metaheuristics*, pages 42–53. Springer, 2005.

[88] Z. Ognjanović, M. Rašković. Some first-order probability logics. *Theoretical Computer Science*, 247(1):191–212, 2000.

[89] Z. Ognjanović, M. Raškovic, Z. Marković. Probability logics. *Zbornik radova, subseries Logic in computer science*, 12(20):35–111, 2009.

[90] C. H. Papadimitriou, K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[91] J. B. Paris. Lecture notes for nonmonotonic logic. *http://www.maths.manchester.ac.uk/∼jeff/*, 2014.

[92] J. Pearl. System Z: A natural ordering of defaults with tractable applications to nonmonotonic reasoning. In *Proceedings of the 3rd Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 121–135. Morgan Kaufmann Publishers Inc., 1990.

[93] J. Pearl et al. *Probabilistic semantics for nonmonotonic reasoning: A survey*. Computer Science Department, University of California, 1989.

[94] C. S. Peirce. *On a improvement in Boole's calculus of logic*. 1867.

[95] M. Pirlot. General local search methods. *Europ. J. Oper. Res.*, 92:493–511, 1996.

[96] R. Poli, J. Kennedy, T. Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.

[97] P. S. Poretskiy. *Solution of the general problem of the theory of probability by using mathematical logic*. 1887.

[98] M. J. Quinn. *Parallel Programming in C with MPI and OpenMP*. McGraw-Hill, 2003.

[99] S. Rana, D. Whitley. Genetic algorithm behavior in the MAXSAT domain. In *Parallel Problem Solving from Nature—PPSN V*, pages 785–794. Springer, 1998.

[100] M. Rašković. Classical logic with some probability operators. *Publ. Inst. Math., Nouv. Sér*, 53(67):1–3, 1993.

[101] M. Rašković, Z. Marković, Z. Ognjanović. A logic with approximate conditional probabilities that can model default reasoning. *International Journal of Approximate Reasoning*, 49(1):52 – 66, 2008.

[102] V. Risch, C. B. Schwind. Tableau-based characterization and theorem proving for default logic. *Journal of Automated Reasoning*, 13(2):223–242, 1994.

[103] A. Robinson. *Non-standard analysis*. Studies in Logic and the Foundations of Mathematics, 1966.

[104] S. J. Russell, P. Norvig. *Artificial intelligence: a modern approach*. Prentice hall Englewood Cliffs, 2003.

[105] T. Schaub, S. Brüning, P. Nicolas. XRay: A prolog technology theorem prover for default reasoning: A system description. *Automated Deduction—Cade-13*, pages 293–297, 1996.

[106] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Chichester, 1986.

[107] C. B. Schwind. A tableaux-based theorem prover for a decidable subset of default logic. In *10th International Conference on Automated Deduction*, pages 528–542. Springer Berlin Heidelberg, 1990.

[108] D. Scott, P. Krauss. Assigning probabilities to logical formulas. *Studies in Logic and the Foundations of Mathematics*, 43:219–264, 1966.

[109] B. Selman, H. A. Kautz, B. Cohen. Noise strategies for improving local search. In *AAAI*, volume 94, pages 337–343, 1994.

[110] W. Spendley, G. R. Hext, F. R. Himsworth. Sequential application of simplex designs in optimisation and evolutionary operation. *Technometrics*, 4(4):441–461, 1962.

[111] T. Stojanović, T. Davidović, Z. Ognjanović. Bee colony optimization for the satisfiability problem in probabilistic logic. *Applied Soft Computing*, 31(0):339 – 347, 2015.

[112] T. Stojanović, A. Kaplarević-Mališić, Z. Ognjanović. An extension of the probability logic LPP_2. *Kragujevac Journal of Mathematics*, 33:45–62, 2010.

[113] T. Stützle, H. Hoos, A. Roli. A review of the literature on local search algorithms for MAX-SAT. *Rapport technique AIDA-01-02, Intellectics Group, Darmstadt University of Technology, Germany*, 2001.

[114] P. Suppes. *Probabilistic inference and the concept of total evidence*. Springer, 1969.

[115] E.-G. Talbi. *Metaheuristics: From Design to Implementation*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2009.

[116] D. Teodorović. Bee colony optimization (BCO). In C. P. Lim, L. C. Jain, S. Dehuri, editors, *Innovations in Swarm Intelligence*, pages 39–60. Springer-Verlag, Berlin Heidelberg, 2009.

[117] D. Teodorović, M. Dell'Orco. Mitigating traffic congestion: solving the ride-matching problem by bee colony optimization. *Transport. Plan. Techn.*, 31:135–152, 2008.

[118] D. Teodorović, M. Šelmić, T Davidović. Bee colony optimization part II: The applications survey. *Yugoslav Journal of Operational Research*, 25(2):185–219, 2015.

[119] M. Villagra, B. Barán. Ant colony optimization with adaptive fitness function for satisfiability testing. *Logic, Language, Information and Computation*, pages 352–361, 2007.

[120] M. Šelmić, D. Teodorović, K. Vukadinović. Locating inspection facilities in traffic networks: an artificial intelligence approach. *Transport. Plan. Techn.*, 33:481–493, 2010.

[121] B. Xing, W.-J. Gao. *Innovative computational intelligence: a rough guide to 134 clever algorithms*. Springer, 2014.

[122] X.-S. Yang. *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[123] X.-S. Yang. *Nature-inspired optimization algorithms*. Elsevier, 2014.

Биографија

Татјана Стојановић (рођ. Тимотијевић), рођена је 1974. године у Крагујевцу. Основну школу и гимназију завршила је Крагујевцу. Природно-математички факултет у Крагујевцу, група математика, смер рачунарство, уписала је 1993/94, а дипломирала 1997. године са просечном оценом 9,26. Од 1998. до 2003. године била је ангажована као асистент-приправник за групу предмета Примењена математика на Природно-математичком факултету у Крагујевцу. Од 2003. године је ангажована, као асистент за ужу научну област Вештачка интелигенција, на истом факултету. Последипломске-магистарске студије на групи математика, смер рачунарство, уписала је школске 1997/98. године и све предмете предвиђене планом и програмом положила са просечном оценом 10. Магистарску тезу под насловом „Доказивачи теорема" одбранила 2002. године и тиме стекла звање магистра информатичких наука.

До сада је била ангажована на реализацији вежби на неколико студијских програма и на неколико предмета и то:

- На студијском програму за стицање звања Дипломирани математичар-информатичар: Рачунарство 1, Алгебра и логика у рачунарству, Вештачка интелигенција и експертни системи;
- На предметима студијског програма за стицање стручног/академског звања Информатичар и Дипломирани информатичар - мастер: Основе програмирања, Софтверски практикум 1, Структуре података и алгоритми 1, Структуре података и алгоритми 2, Формални језици, аутомати и језички процесори, Теоријско рачунарство, Представљање знања и закључивање, Методика програмирања;
- На другим студијским групама и факултетима: Основи биостатистике и рачунарства – на групи за биологију, Математика – на групи за хемију, Математика 2 – на групи за физику, Математика 1, Математика 2 – на Машинском факултету у Крагујевцу, Информатика – на Учитељском факултету у Јагодини.

Учествовала је у реализацији 2 научна, 2 техничко-технолошка као и 2 стручна пројекта. Активно је учествовала у раду семинара из Логике на Математичком институту Српске академије наука и уметности.

Од 2004. до 2008. године била је секретар часописа Kragujevac Journal of Mathematics.

Активни је члан Друштва математичара Србије. Од 2003. до 2006. године члан је редакције Тангенте, часописа за математику и рачунарство намењеног ученицима

средње школе у издању Друштва математичара Србије. Од 2006. године члан је редакције Математичког листа, часописа за математику и рачунарство намењеног ученицима основне школе у издању Друштва математичара Србије. Активан је предавач у оквиру Математичке радионице младих, која од 1999. године ради са младим талентима у Крагујевцу. Од 2006. до 2012. године председник Републичке комисије за Такмичења из програмирања за ученике основних школа. Године 2007. била је члан жирија на Јуниорској Балканској Олимпијади из информатике, а 2008. године лидер тима Републике Србије на Јуниорској Балканској Олимпијади из информатике. У периоду од 2003. до 2005. била је ментор ученицима основних школа на изради радова у оквиру активности Регионалног центра за таленте.

Резултате својих истраживања објавила је у оквиру следећих научних радова:

1. T. Stojanović, T. Davidović, Z. Ognjanović, *Bee colony optimization for the satisfiability problem in probabilistic logic*, Applied Soft Computing 31 (2015): 339-347
   ISSN: 1450-9628 (M21)

2. T. Stojanović, A. Kaplarević-Mališić, Z. Ognjanović, *An extension of the probability logic LPP_2*, Kragujevac J. Math. 33 (2010): 45-62
   ISSN: 1450-9628 (M51)

3. M. Mosurović, T. Stojanović, A. Kaplarević-Mališić, *Reasoning in Basic Description Logics and Description Logics with Modal Operators*, Zbornik radova Logic in Computer Science, Matematički institut SANU (2009): 113-158
   ISBN: 978-86-80593-40-1 (M45)

4. Z. Ognjanović, T. Timotijević, *On two approaches to modal theorem proving*, Novi Sad J. Math., vol 30, No. 2 (2000): 83-93
   ISSN: 1450-5444 (M53)

5. T. Timotijević, *One implementation of PL prover algorithm*, Kragujevac J. Math. 23 (2000): 119-130
   ISSN: 1450-9628 (M53)

Добијене резултате излагала је на следећим конференцијама:

- T. Stojanović, N. Ikodinović, T. Davidović, Z. Ognjanović, *Dealing with satisfiability problem in default logic using Bee-colony optimization*, Četvrta nacionalna konferencija „Verovatnosne logike i primene“, Beograd, 02.10.2014.

- T. Stojanović, T. Davidović, Z. Ognjanović, *Bee-colony optimization for the satisfiability problem in probabilistic logic*, Treća nacionalna konferencija „Verovatnosne logike i primene“, Beograd, 26.09.2013.

- T. Stojanović, Probability description language $P - \mathcal{ALCN}$, Prva nacionalna konferencija „Verovatnosne logike i primene“, Beograd, 29.-30.09.2011.

- D. Stefanović, S. Radisavqević, T. Timotijević, A. Kaplarević, *Tehnički informacioni sistem reyervnih delova vozila*, Info-Teh 2001, XV naučno-stručni skup, Zbornik radova, 316-319, Vrnjačka Banja, 18-22 jun 2001.

- Z. Ognjanović, T. Timotijević, *On two approaches to modal theorem proving*, 10. Kongres matematičara jugoslavije, Beograd, 21-24. januar 2001
- T. Timotijević, *One implementation of PL prover algorithm*, 10. Kongres matematičara jugoslavije, Beograd, 21-24. januar 2001
- Z. Ognjanović, T. Timotijević, *On two approaches to modal theorem proving*, III International Conference on Theoretical Aspects of computer Science with practical applications, Novi Sad, Yugoslavia, September 6-7, 2000

Коаутор је једне књиге

- M. Čabarkapa, S. Matković, T. Timotijevć, *Zbirka zadataka iz programiranja – okružna i republička takmičenja učenika osnonih škola 1988-2006*, Beograd, 2007

Applied Soft Computing 31 (2015) 339–347

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Bee colony optimization for the satisfiability problem in probabilistic logic

Tatjana Stojanović [a,*], Tatjana Davidović [b], Zoran Ognjanović [b]

[a] Faculty of Science, University of Kragujevac, R. Domanovića 12, 34000 Kragujevac, Serbia
[b] Mathematical Institute, Serbian Academy of Science and Arts, Kneza Mihaila 36 (P.O. Box 367), 11001 Belgrade, Serbia

ARTICLE INFO

Article history:
Received 14 October 2013
Received in revised form 15 October 2014
Accepted 8 March 2015
Available online 17 March 2015

MSC:
03B60
68T20
68T27

Keywords:
Probabilistic satisfiability
Conditional probability
Approximate probability
Meta-heuristics
Swarm intelligence

ABSTRACT

This paper presents the first heuristic method for solving the satisfiability problem in the logic with approximate conditional probabilities. This logic is very suitable for representing and reasoning with uncertain knowledge and for modeling default reasoning. The solution space consists of variables, which are arrays of 0 and 1 and the associated probabilities. These probabilities belong to a recursive non-Archimedean Hardy field which contains all rational functions of a fixed positive infinitesimal. Our method is based on the bee colony optimizationmeta-heuristic. The proposed procedure chooses variables from the solution space and determines their probabilities combining some other fast heuristics for solving the obtained linear system of inequalities. Experimental evaluation shows a high percentage of success in proving the satisfiability of randomly generated formulas. We have also showed great advantage in using a heuristic approach compared to standard linear solver.

1. Introduction

Working with uncertain knowledge has been well documented problem in mathematical logic and computer science, since the first works of Leibnitz and Bool. The unique solution to this problem has not been found yet, but there are many ideas and different variants of solutions for particular problems, that are used in artificial intelligence. Many of the formalisms for representing and reasoning with uncertainty are based on probabilistic logics [1–10]. These logics are extensions of classical logic with probabilistic operators. Satisfiability problem in these logics (PSAT) can be reduced to linear programming problem. However, solving it by any standard linear solver is inapplicable in practice due to the complexity of the problem. For example, the application of Fourier–Motzkin elimination procedure yields the exponential growth in the number of inequalities in the system. Therefore, the application of some other techniques for solving this problem, such as different types of meta-heuristics, could prove very useful.

Using meta-heuristics for solving satisfiability problems is not a new idea. The most interesting problems in propositional logic are satisfiability problem (SAT) and maximum satisfiability problem (MAX-SAT), i.e., the problem of determining the maximum number of clauses of a given Boolean formula in conjunctive normal form, that can be made true by an assignment of truth values to the variables. Several methods based on different heuristics have been developed for SAT and MAX-SAT. Many of those methods are based on local search procedure and some of them are presented in [11–14]. Heuristics based on swarm intelligence, like Ant Colony Optimization (ACO) or Bee Swarm Optimization (BSO), were also applied to SAT [15,16]. For this type of problem probabilistic approach is presented in [17]. Genetic Algorithm (GA) is another approach used for dealing with SAT and/or MAX-SAT [18,19], and it is also combined with some other heuristics [20]. In probabilistic logics presented by Nilson in [21], Fagin et al. [1] or by Rašković et al. [2], local search based heuristics [22], Tabu Search (TS) [23], GA [24,25], Variable Neighbourhood Search (VNS) [26], and combination of GA and VNS [27] were used for solving PSAT.

Here, we discus the satisfiability problem in approximate conditional probabilities logic described by Rašković et al. in [4]. We denote this version of satisfiability problem with CPSAT-ε. The main differences between PSAT and CPSAT-ε are:

- CPSAT-ε involves conditional probability operator on the contrary to PSAT.
- Probabilities of formulas in CPSAT-ε may take infinitesimal values, and not only real-values as in PSAT.

The first use of infinitesimal was by Leibniz, when he introduced differentials. Later, Robinson [28] showed how infinitely large and infinitesimal numbers can be rigorously defined and used. Characteristics of CPSAT-ε, given above, do not allow us to use the existing methods for PSAT that work with formulas containing real-valued unconditional probabilities only.

The logic, described in [4], enriches the propositional calculus with probabilistic operators which are applied to propositional formulas: $CP_{\geqslant s}(\alpha, \beta)$, $CP_{\leqslant s}(\alpha, \beta)$ and $CP_{\approx s}(\alpha, \beta)$, with the intended meaning that the conditional probability of α given β is "at least s", "at most s" and "approximately s", respectively. This way of knowledge representation and reasoning can be widely used. One of the most obvious examples would be the process of reaching diagnosis in medicine. Occurrence of some symptoms with adequate certainty represented in percents, leads to conclusion that a

* Corresponding author. Tel.: +381 642183742.
E-mail addresses: tanjat@kg.ac.rs (T. Stojanović), tanjad@mi.sanu.ac.rs (T. Davidović), zorano@mi.sanu.ac.rs (Z. Ognjanović).

http://dx.doi.org/10.1016/j.asoc.2015.03.017
1568-4946/

[3] M. Borisavljević, *A connection between cut elimination and normalization*, Archive for Mathematical Logic, vol. **45**, (2006), 113-148

[4] M. Borisavljević, *Normal derivations and sequent derivations*, Journal of Philosophical Logic, Vol. **37**, No. 6, (2009), 521-548

[5] M. Borisavljević, *Maximum segments in extended natural deduction*, Publications de l'Institut Mathematique, to appear

[6] G. Gentzen, *Untersuchungen über das logische Schließen*, Mathematische Zeitschrift **39**, (1935) 176-210, 405-431 (English translation in [Gentzen 1969])

[7] G. Gentzen, *The Collected Papers of Gerhard Gentzen*, M. E. Szabo (ed.), North-Holland, 1969

[8] G. E. Minc, *Normal forms for sequent derivations* In Odifreddi, P. (ed.) Kreiseliana: About and Around Georg Kreisel, Peters (1996), 469-492

[9] J. von Plato, *Natural deduction with general elimination rules*, Archive for Mathematical Logic vol. **40/1**, (2001), 541-567

[10] G. Pottinger, *Normalization as a homomorphic image of cut elimination*, Annals of Pure and Applied Logic vol. **12**, (1977), 323-357

[11] D. Prawitz, *Natural Deduction*, Almquist and Wiksell, Stockholm, 1965

[12] D. Prawitz, *Ideas and results in proof theory*, in: J. E. Fenstad, (ed.) *Proc. of the Second Scandinavian Logic Symposium*, Norht-Holland, 1971, 235-307

[13] P. Schrdeder-Heister, *A natural extension of natural deduction*, The Journal of Symbolic Logic vol. **49/4**, (1984), 1284-1300

[14] J. Zucker, *The correspondence between cut-elimination and normalization*, Annals of Mathematical Logic vol. **7**, (1974), 1-112

Kragujevac J. Math. 33 (2010) 45–62.

AN EXTENSION OF THE PROBABILITY LOGIC LPP_2

Tatjana Stojanović[1], Ana Kaplarević-Mališić[1] and Zoran Ognjanović[2]

[1] *University of Kragujevac, Faculty of Science, R. Domanovića 12, 34000 Kragujevac, Serbia*
(e-mails: tanjat@kg.ac.rs, ana@kg.ac.rs)

[2] *Mathematical Institute SANU, Kneza Mihaila 36, 11000 Belgrade, Serbia*
(e-mail: zorano@mi.sanu.ac.rs)

(Received September 25, 2009)

Abstract. In this paper we present a probability logic which allows Boolean combinations of formulas of the form of $a_1w(\varphi_1)+\cdots+a_kw(\varphi_k) \geq c$, where $\varphi_1,\ldots,\varphi_k$ are propositional formulas, $a_1,\ldots,a_k,c$ are rational numbers. The logic with such syntax was introduced in [1]. We present axiomatic system along with the ideas from [2, 3] and prove extended completeness, instead of simple completeness shown in [1].

2010 Mathematics Subject Classification: 03B48, 60A05.

Key words: Probability logic, Probability model, Linear inequalities involving probabilities.

1. INTRODUCTION

In the past two decades, formalisms for representing uncertain information and knowledge and reasoning about them are the subject of increasing interest in many fields od science, starting with theoretical computer science, through artificial intelligence as first consumer of the research results in that field, distributed systems,